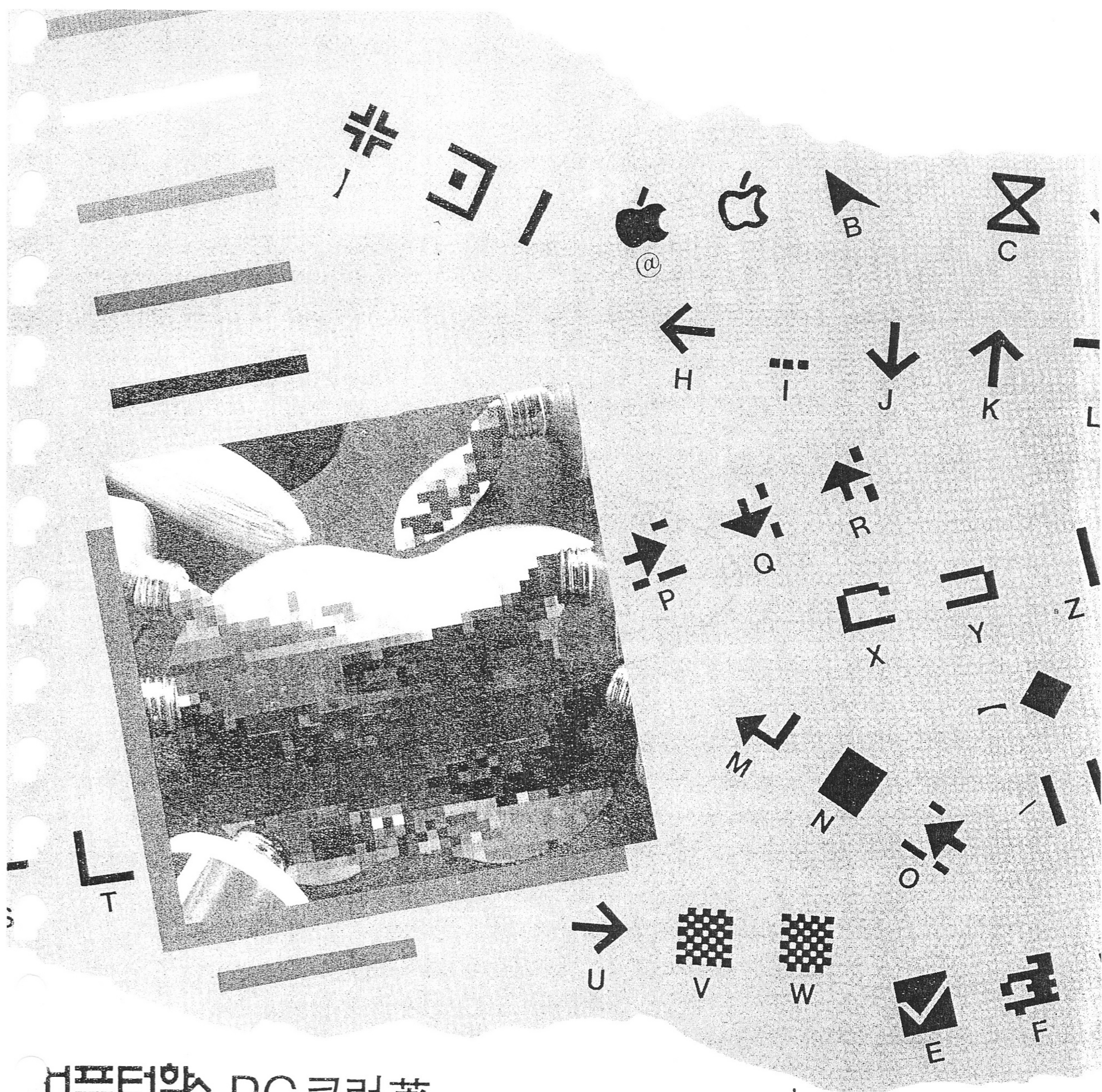


컴퓨터학습 PC클럽著

애플IIe 테크노트

APPLE IIe TECH-NOTE

차　례

제1장 애플Ⅱe 란

제2장 애플Ⅱe 의 하드웨어 개관

제3장 CPU 및 기계어

제4장 128K의 관리와 사용

제5장 Ⅱe 의 그래픽과 사운드

제6장 기타 애플 Ⅱe 의 특징

제7장 애플의 표준 OS인 ProDOS 및 ProDOS BASIC

제8장 애플Ⅱe 의 통합 소프트웨어– AppleWorks

제9장 애플 Ⅱe 용 언어 프로그램

부록. 애플 Ⅱe 의 응용

제 1 장

애플 IIe란

§1. 애플 IIe가 태어나기까지
§2. 애플 IIe의 시스템 구성



§1. 애플Ⅱe가 태어나기까지

애플Ⅱe를 살펴보기에 앞서 애플Ⅱe가 만들어지게 된 애플컴퓨터의 역사를 살펴보기로 하자. 애플컴퓨터가 태어난 지도 올해로 13년째 접어들고 있다. 여기서는 애플컴퓨터의 설계자이고 애플사의 공동 설립자 중의 한 사람인 스티브 워즈니악(Steve Wozniak)을 중심으로 살펴보자.

스티브 워즈니악은 이미 고등학교 시절에 여러 대의 컴퓨터를 설계하는 등 컴퓨터에 남다른 재능을 갖고 있는 실력파였다. 그후 학교를 졸업하고 휴렛팩커드사에서 엔지니어로 일하면서 더 뛰어난 설계 전문 기술을 익히게 되었다. 컴퓨터 설계에 뛰어난 기술을 발휘하던 스티브는 마침내 자신이 원하는 컴퓨터를 직접 설계하기로 하였다. 휴렛팩커드에 있는 그의 연구실에는 9830이라는 베이직이 작동하는 탁상용 컴퓨터가 있었다. 이 컴퓨터는 베이직을 곧바로 작동시킬 수 있었는데, 그것이 애플Ⅰ의 목표였다. 그러나 자금이 부족한 관계로 값이 싼 부품을 사용하지 않으면 안되었다. 따라서 마이크로 프로세서는 6502를 택하고 4K스태틱 램 보드를 이용한 애플Ⅰ을 만들어냈다. 그리고 친구인 스티브 잡스(Steve Jobs)와 회사를 설립하기로 한 스티브 워즈니악은 자신의 자동차인 폭스바겐과 HP 계산기를 팔아서 자금을 조달했다. 이렇게 제작된 최초의 애플컴퓨터는 한 대당 5백 달러, 소매 가격은 6백66달러에 판매를 하였으며 1년여 만에 1백75대가 팔렸다.

당시의 애플I의 특성을 요약하면 다음과 같다. CPU는 6502를 사용했으며 램의 용량은 8K에 불과하였다. 베이직이 4K의 메모리에 로드되고 직접 베이직 프로그램을 짜는 데에는 나머지 4K를 사용하도록 하였다. 완전히 조립된 보드로 공급하였으며 비디오 커넥터가 있었으나 비디오 모니터는 별도로 자신의 것을 연결하도록 하였다. 키보드도 있어야 했으며 16핀의 DIP 커넥터에 연결해야 했다. 전원장치는 보드 위에 설치했지만 5볼트와 12볼트의 트랜스 두 개를 접속시키게 되어 있었다. 스피커도 없었고 그래픽이나 컬러 기능도 없었다. 단지 초당 60자의 텍스트만을 디스플레이시킬 뿐이었다. 그리고 비디오 터미널을 내장하여 키보드 입력을 받아서 비디오 터미널에 표시시키고 4K 메모리의 뱅크1과 뱅크2의 어드레스들을 손으로 직접 연결시켜야 했다. 한 개의 외부 카드 커넥터가 있었으며 곧바로 카세트 인터페이스 카드가 접속되었다. 또한 보조 커넥터가 있어서 이론상으로는 나중에 필요한 보조 장치를 추가로 확장시킬 수 있었다. 이것이 애플I의 모든 것이다. 이것은 그 당시 큰 인기를 끌었고 워즈니악은 잡스와 공식적인 동업 관계를 맺고 1977년 애플사를 창립했다.

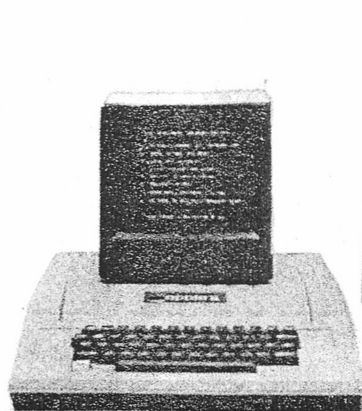
그리고 컴퓨터의 이름을 애플이라고 명명한 것은 스티브 잡스의 머리에서 나온 것인데, 어떻게 유래한 것인지는 지금까지 밝히지 않고 있다. 워즈니악의 이야기로는 잡스에게는 영적 감각이 있어서 그 때문에 우연히 떠오른 것이라 한다. 그리고 다른 더 좋은 이름을 생각해내려 해도 그 이상 좋은 이름이 없어 애플로 정했다고 한다.

애플Ⅰ의 제작에 용기를 얻은 두 사람은 여러가지 기능을 추가시킨 애플Ⅱ를 발표하여 세상을 다시

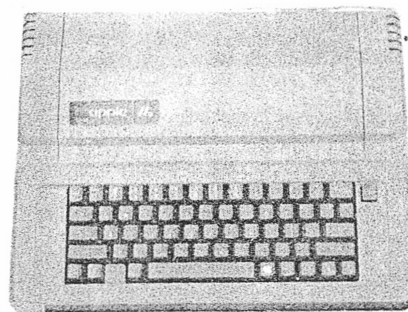
한번 놀라게 하였다. 이 애플Ⅱ는 6K 정수 베이직을 내장하고 4K 램을 내장하고 있으며 컬러 그래픽 기능도 내장시켰다. 지금의 상황에서 본다면 형편없는 것이지만 당시로서는 대단한 것이었다. 그때는 정수 베이직만 사용하는 것으로도 장점이 되었으나 문제점도 드러났다. 그것은 실수(real number)를 사용하지 못한다는 것과 정수 베이직에는 고해상 그래픽 명령어가 들어있지 않다는 것이다. 따라서 이러한 결점을 보완하기 위해서 또 다른 베이직을 만들었는데, 그것이 애플소프트라고 하는 베이직이다. 애플 소프트 베이직은 초기에 카세트 테이프에 수록하여 공급하였지만 여러가지 문제점으로 개량형을 발표했는데, 이것이 애플소프트Ⅱ라고 부르는 현재의 10K ROM 베이직이다(이후 애플소프트Ⅱ를 간단히 애플소프트라고 부르고 있다). 이것은 마이크로소프트(Microsoft)사가 제작한 것으로 롬 카드(ROM-card) 로 공급하여 베이직에 관한 모든 문제를 해결하였다.

1978년 애플Ⅱ에 플로피디스크 드라이브를 부착한 제품이 등장하였다. 애플사는 SHUGART사 제품인 SA400 드라이브를 사용해서 기억 용량이 116KB이고 전송 속도가 156K비트 / 초인 미니 디스크를 만들어서 DISK Ⅱ라 명명하여 공급했다. 당시 다양한 기능과 좋은 성능을 자랑하던 애플Ⅱ도 1978년에 개량형 모델을 내놓았는데, 개량된 점은 다음과 같다. 전원을 켜는 것과 동시에 리셋되고 텍스트 모드일 때는 컬러 신호가 완전히 흑백 비디오로 표시되고 고해상 컬러 그래픽 모드에서는 여섯 가지 색으로 표시된다(이전은 4색). 이밖에도 카세트 인터페이스 회로도 약간의 변경이 있었다.

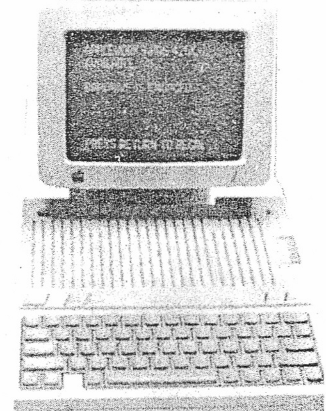
애플Ⅱ에 만족하지 않고 또 다른 하나의 계획에 착수하여 1979년 가을에 애플Ⅱ+가 발표되었다. 애플Ⅱ+는 오토 스타트 롬과 애플소프트라는 실수 베이직을 내장하고 베이직으로도 그래픽을 할 수 있었다. 바로 이것이 우리나라에 널리 보급된 애플 기종이다. 애플Ⅱ+의 발표로 애플사는 급성장했고 스티브 워즈니악은 퍼스널 컴퓨터 분야에 대한 공로가 인정되어 그레이스 머레이 호퍼상을 수상했다. 그 뒤로 애플사는 4년간의 프로젝트 하에 1983년 기존 애플Ⅱ+와는 상대도 안될 정도로 엄청난 하드



애플Ⅱ+



애플Ⅱe

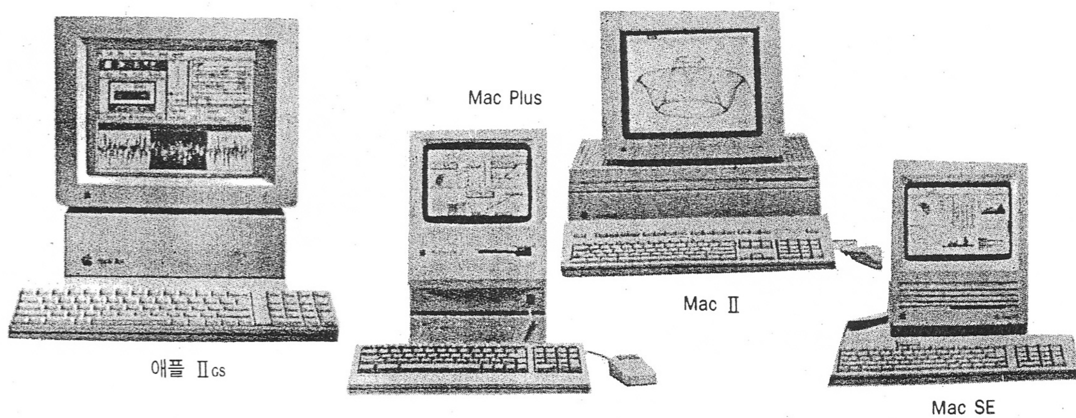


애플Ⅱc

웨어적 변경을 한 애플Ⅱe를 발표하였다(특성은 이 책에서 따로 다루고 있음). 애플Ⅱe는 기존의 애플Ⅱ나 애플Ⅱ+의 한계를 뛰어넘으면서도 호환성을 유지한 걸작이었다. 그 뒤로 애플사는 애플Ⅱe에 대한 유저들의 호응에 보답하고자 1985년에 Enhanced 애플Ⅱe를 내놓았다. 이것은 Ⅱc에 사용되었던 65C02를 사용하고 있으며 마우스용 문자 및 베이직에 약간의 개선이 있었다.

연도	기 종	언 어	DOS	CPU	기 타
1976	Apple Ⅰ	카세트 베이직	·	6502	잡스와 워즈니악이 만듬
1977	Apple Ⅱ	정수베이직 내장	·	6502	애플사 창립
1978	·	·	DOS 3.1	Auto Start ROM	디스크Ⅱ 처음 도입
1979	Apple Ⅱ+	애플소프트 내장	DOS 3.2	6502	·
1980	애플 Ⅲ	·	DOS 3.3	·	애플 Ⅱ 시스템 발매
1981	·	·	·	·	하드디스크 Profile 도입
1983	Lisa Apple Ⅱe	· 애플소프트 프로도스 베이직	· ProDOS	· 6502A	ProDOS 발표
1984	Macintosh Apple Ⅱc MAC 512K	– · –	– · –	– 65C02 –	– 애플웍스 발표 –
1985	애플 Ⅱe enhanced			65C02	Laser-writer Apple-talk Lisa→MacintoshXL 워즈니악, 잡스 애플사를 떠남
1986	Apple Ⅱ$_{GS}$ Apple Ⅱc enhanced	–	ProDOS 8 ProDOS 16	65C816	

표 1. 1 애플사의 역사 및 기종



애플컴퓨터 시리즈가 발표되기까지는 워즈니악과 잡스의 피나는 노력과 연구가 있었다. 워즈니악의 취미로 비롯되어 설립된 애플사는 Ⅱe 이후로 IIc, III, LISA, Macintosh 등등 여러 기종들을 개발하였다. 여기서는 이 책의 취지에 맞도록 Ⅱe까지만 살펴보았다. 애플Ⅱe는 애플사의 대표라고 할 정도로 지금까지 많은 사람들의 인기를 누리고 있고 앞으로도 그 인기는 계속될 것이다. 끝으로 애플사의 역사를 도표로 작성하였으므로 참고하기 바란다.

§ 2. 애플 Ⅱe의 시스템 구성

1983년에 처음 등장하여 현재 미국 내의 8비트 PC 중 최다 보급률을 자랑하는 애플Ⅱe는 1985년을 기준으로 Original Ⅱe와 Enhanced Ⅱe로 나누어진다. Enhanced Ⅱe는 1984년에 등장한 Ⅱc의 새로운 기능들을 보강시켰다. 이 책에서도 이에 발맞추어 Enhanced Ⅱe를 기준으로 설명한다. 그래서 이후로는 Enhanced Ⅱe를 Ⅱe라 부른다. 그리고 현재 우리나라의 Ⅱe 호환 기종도 Enhanced Ⅱe이다.

(1) CPU(중앙처리장치)

65C02(CMOS형의 1MHz)를 기본으로 유저가 원하는 6502 계열의 8비트 마이크로 프로세서를 어느 것이든지 사용할 수 있다. 해당되는 프로세서는 다음과 같다.

6502(1MHz), 6502A(2MHz), 6502B(3MHz), 65C02(4MHz)

CMOS형의 CPU가 기본으로 사용된 것은 그 나름대로의 장점이 있다. 우선 전력 소모가 적고 노이즈의 영향을 덜 받으며 신뢰도가 높다. 그리고 10개의 기계어 명령이 추가되어 있어 66개의 명령을 수행할 수 있고 어드레싱 모드도 두 개나 추가되었다.

(2) ROM

Ⅱe용의 시스템 모니터(Ⅱ+에는 없는 기능들이 추가되어 있다), 애플소프트 베이직 인터프리터, 80컬럼 표시 루틴, 자체 검사 기능을 포함하여 16K를 2개의 8K ROM에 내장했다. 단 국산 Ⅱe는 16K롬(27128) 하나를 사용했다.

참고 Ⅱe 베이직에서는 약간의 개선이 있었는데 그 사항은 소문자로 베이직 프로그램을 작성하는 것이 허용되는 것과 프로그램이 리스트될 때 문번호 앞의 한 칸을 비워 놓아 프로그램을 수정할 때 편리하게 한 점이다.

(3) RAM

기본 64K 내장(8K DRAM 8개), 보조 메모리 64K를 활용하여 128K 사용 이 가능하다. 128K의 메모리 관리는 여러 개의 용도별 블럭으로 나누어져서 사용자가 필요에 따라 원하는 부분만 선택하여

관리하게 된다. 선택은 거의 소프트 스위치로 하게끔 되어 있다. 80컬럼 모드가 작동되지 않은 상태에서는 16K 랭귀지 카드가 설치된 Ⅱ+와 비슷하다.

(4) 키보드

영문 대·소문자와 각종 특수 문자를 포함한 아스키 문자 세트 전부와 특수 기능키 등 총 63키이다. 여기에는 Ⅱ+에는 없는 키가 추가되어 있는데 다음과 같다.

Caps Lock

알파벳키들이 모두 소문자 기능을 가지게 되었으므로 필수적으로 추가된 키이다. 이 키는 한 번 눌러주면 눌러진 채로 있으며, 다시 한 번 눌러야 원위치로 복귀된다. 키가 눌러진 상태에서는 SHIFT 키의 상태에 관계없이 모든 알파벳키는 대문자로만 입력된다. 그러나 키가 눌러지지 않은 상태에서는 알파벳키를 그냥 누르면 소문자, SHIFT키와 함께 누르면 대문자로 입력된다.

TAB

커서를 일정한 칸 수만큼 이동시키는 기능을 가지고 있으나 본체 안에는 기능이 없고 코드만 발생시킨다. 따라서 실제의 이동은 사용하는 프로그램에 따라서 모두 다르며 사용되지 않을 수도 있다. 실제로 베이직 상에서는 전혀 동작하지 않는다.

DELETE

커서 왼쪽의 한 문자를 삭제하는 기능을 가지고 있으나 역시 코드만 발생시킨다. 따라서 TAB키와 마찬가지로 프로그램에 따라서 사용법이 다르고 사용되지 않을 수도 있다. 베이직상에서는 커서 모양을 그대로 복사하고 오른쪽으로 한 칸 이동한다.

상하 이동키 Open Apple키와 Solid Apple키

어떤 컴퓨터에서도 찾아볼 수 없는 독특한 키 두 개가 추가되었다. 스페이스바 양쪽에 사과 모양을 한 키가 하나씩 있는데 왼쪽에 있는 것을 Open Apple키, 오른쪽에 있는 것을 Solid Apple키라고 한다. 이 두 개의 키에는 독자적인 기능은 없다. 그러나 Open Apple은 조이스틱의 0번 스위치에, Solid Apple은 1번 스위치에 연결되어 있다.

강화된 기능

Auto Repeat 기능

N-key Rollover : 가장 나중에 입력된 키가 입력이 된다.

(5) 사용 문자

• 40컬럼 모드

대문자, 특수문자 : Normal, Inverse, Flash

소문자 : Normal

Mouse Text라는 특수 그래픽 문자를 사용할 수 있다.

• 80컬럼 모드

대문자, 소문자, 특수문자 : Normal, Inverse

Mouse Text

Flash 기능은 없음.

이 80컬럼 모드는 Ⅱ+에서도 바이덱스 카드라는 것에 의해서 가능했었다. 하지만 그 카드는 애플사에서 내놓은 표준 카드가 아니었다. 그래서Ⅱe에서는 애플사에서 내놓은 80컬럼 카드를 보조 슬롯에 꽂아 사용하도록 되어 있다. 우리나라의 Ⅱe 호환 기종의 경우처럼 80컬럼 기능이 내장되어 있을 경우에도 PR#3이라는 명령으로 80컬럼을 작동시킨다. 이 80컬럼은 Ⅱ+에서의 80컬럼처럼 따로 비디오 램을 가지고 있는 것이 아니라 보조 메모리를 사용한다. 즉 보조 메모리 가운데 $400~$7FF 부분과 메인 메모리의 $400~$7FF 부분을 비디오 램으로 사용하는 것이다. 간단히 설명해서 80컬럼에 APPLE을 표시한다고 하면 메인 메모리에는 A, P, E가, 보조 메모리에는 P, L이 저장되는 것이다. 80컬럼 모드가 되면 40컬럼 모드에서의 체커보드 모양의 커서는 반짝이지 않는 사각형으로 바뀐다. 그리고 ESC키가 눌러졌을 때는 반전된 +가 표시되므로 ESC가 눌러졌다는 것을 확실히 알 수 있다. 그리고 Ⅱe의 80컬럼에서는 커서 위치 지정 및 윈도우 지정까지 80컬럼 화면을 기준으로 적용된다.

(6) 화면 표시 모드

40컬럼 텍스트 모드(40자*24줄)

80컬럼 텍스트 모드(80자*24줄, 40자*24줄 전환 가능)

GR 모드(수평 40*수직 48블럭, 16색)

더블 고해상도 GR 모드(수평 80*수직 48블럭, 16색)

HGR 모드(수평 80*수직 192도트, 6색)

더블 고해상도 HGR 모드(수평 560*수직 192도트, 모노크롬)

(수평 140*수직 192도트, 16색, 색 간섭 때문)

위 네 가지 그래픽 모드에서는 텍스트와의 4줄 혼합 화면 구성이 가능하다. 더블 고해상도 HGR을 사용할 때의 비디오 램 메모리 맵은 메인 메모리와 보조 메모리를 한 바이트씩 교대로 사용하고 있다. HGR에서는 각 바이트의 최상위 비트가 컬러를 나타냈지만 더블 고해상도 HGR에서는 최상위 비트가 사용되지 않고 4비트씩 묶어서 그 조합에 의해 16색을 표현한다.

(7) 입출력 장치

키보드 입력 : 키보드 접속용 24핀 커넥터

숫자 키 패드 접속용 11핀 커넥터

비디오 출력 : RCA형 비디오 출력 잭
4핀 내장 비디오 커넥터
게임 I/O : 16핀 게임 I/O 커넥터와 9핀 D형 커넥터가 있는데 9핀 D형 커넥터는 16핀 커넥터에서 불필요한 핀들이 제거되어 있다.
스피커 : 0.5W 8Ω
카세트 : 입력 잭, 출력 잭
전 원 : 사용 가능한 전압 107V~132V AC
최대 전력 60와트(최고 80와트)
공급 전류 +5V 2.5A
+12V 1.5A
−5V 250mA
−12 250mA
슬 롯 : 50핀 범용 슬롯 7개, 60핀 보조 슬롯 1개

(8) 기타

Ⅱe는 커스텀 칩의 이용으로 칩 수가 31개로 대폭 줄어들었다. 그리고 커스텀 칩을 테스트할 수 있는 자체 검사 기능이 있다. 이 자체 검사 기능으로 하여 커스텀 칩인 IOU(Input/Output Unit : 입출력 기능 관리), MMU(Memory Management Unit : 메모리 관리 기능)를 비롯하여 제로페이지, 램 상황을 체크할 수 있다. 시동은 CTRL−Solid Apple−RESET을 누른다. 그러면 10초 뒤에 메시지가 나오는데 System OK라고 나오면 정상이다. 그밖에는 다음과 같다.

MMU : 메모리 관리 기능 이상
IOU : 입출력 관리 기능 이상
RAMZP : 제로페이지의 메모리 이상
RAM : 제로페이지 이외의 메모리 이상

이상에서 보는 바와 마찬가지로 애플Ⅱe의 시스템 구성은 Ⅱ+보다는 월등히 우수하고 잘 되어 있다.

제 2 장

애플 Ⅱe의 하드웨어 개관

§ 1. 시스템 하드웨어

1. Enhanced 애플 Ⅱe

미국에서 1983년에 애플 Ⅱe(Original이라 부름)가 나온 지 2년 뒤인 1985년에 개량형인 Enhanced 애플 Ⅱe(Enhanced라 부름)가 나왔다.

Ⅱe의 결점을 보완한 Enhanced 애플 Ⅱe는 아래와 같은 특징이 있다.

• 오리지널 Ⅱe는 전원을 켤 때 Apple][라고 표시되나 개량형은 Apple //e로 표시된다. 그리고 모니터 루틴 안의 $FBC0의 내용도 다르다. 오리지널은 $FBC0 : EA(십진수 : 234)라고 나오나 개량형은 $FBC0 : E0(십진수 : 224)로 표시된다.

• 80컬럼을 HTAB, TAB, SPC, PRINT 명령들이 지원해 준다.

• 개량형은 소문자로도 프로그램을 입력시킨 후 리스트시키면 자동적으로 대문자로 표시된다. 하지만 DATA, REM문 등에는 적용되지 않는다.

• 시스템 모니터에서도 소문자 입력과 ASCII 입력이 가능하다. 또한 모니터 검색 기능이 있고 미니 어셈블러를 내장하고 있다.

2. 애플 Ⅱe의 Circuit Board의 주요부

• CPU : Central Processing Unit
• Keyboard Encoder
• Keyboard ROM
• Two Interpreter ROM
• Video ROM
• Custom Chip IOU : Input/Output Unit
  MMU : Memory Management Unit
  PAL : Program Array Logic

(1) IOU(입출력 유니트)

• IOU 핀 번호도

GND	1	40	H0
GR	2	39	SYNC'
SEGA	3	38	WNDW'
SEGB	4	37	CLRGAT'
VC	5	36	RA10'
80VID'	6	35	RA9'
CASSO	7	34	VID6
SPKR	8	33	VID7
MD7	9	32	KSTRB
AN0	10	31	AKD
AN1	11	30	C0xx
AN2	12	29	A6
AN3	13	28	+5V
R / W'	14	27	Q3
RESET'	15	26	φ0
(n.c.)	16	25	PRAS'
RA0	17	24	RA7
RA1	18	23	RA6
RA2	19	22	RA5
RA3	20	21	RA4

• IOU 내의 각 핀들의 기능

핀번호	핀이름	기 능
1	GND	접지
2	GR	그래픽 모드 실행
3	SEGA	텍스트 모드상에서 하위 번지의 캐릭터 설정을 위해 VC와 SEGB 실행
4	SEGB	텍스트 모드상에서 VC와 SEGA 실행. 그래픽 모드상에서 저해상도와 고해상도의 선택
5	VC	수직 카운터 비트를 화면에 표시. 텍스트 모드상에서SEGA, SEGB, VC는 표시될 8열의 문자 도트 패턴을 결정. 저해상 모드에서는 1바이트에서 정의된 상·하위 블럭을 선택.
6	80VID'	80컬럼 실행
7	CASSO	카세트 출력 신호
8	SPKR	스피커 출력 신호
9	MD7	비트7의 내부 IOU 플래그
10~13	AN0~AN3	어눈시에이터의 출력
14	R / W'	65C02 읽기 쓰기 제어 신호
15	RESET'	전원을 켜고 출력을 리셋한다.
16	–	연결되지 않았음
17~24	RA0~RA7	다중화된 RAM 번지를 재충전(phase 1)
25	PRAS'	하위 번지 스트로브(phase 0)
26	φ0	마스터 클럭 Phase를 0으로
27	Q3	타이밍 신호의 중개
28	+5V	전원
29	A6	65C02의 비트6 번지
30	C0xx	I / O번지 실행
31	AKD	Any-Key-Down신호
32	KSTRB	키보드 스트로브 신호
33, 34	VIDD7,VIDD6	영상 출력 데이타 비트
35, 36	RA9', RA10'	영상 출력 제어 비트
37	CLRGAT'	컬러 게이트
38	WNDW'	공백 영상 신호
39	SYNC'	동기화된 영상 신호
40	H0	수평 타이밍 영상 신호

(2) MMU(메모리 관리 유니트)

• MMU 핀 번호도

GND	1	40	A1
A0	2	39	A2
φ0	3	38	A3
Q3	4	37	A4
PRAS'	5	36	A5
RA0	6	35	A6
RA1	7	34	A7
RA2	8	33	A8
RA3	9	32	A9
RA4	10	31	A10
RA5	11	30	A11
RA6	12	29	A12
RA7	13	28	A13
R / W'	14	27	A14
INH'	15	26	A15
DMA'	16	25	+5V
EN80'	17	24	Cxxx
KBD'	18	23	RAMEN'
ROMEN2'	19	22	R / W' 245
ROMEN1'	20	21	MD7

• MMU 내의 각 핀들의 기능

핀번호	핀이름	기 능
1	GND	접지
2	A0	65C02 어드레스 입력
3	φ0	Phase0 클럭신호 입력
4	Q3	타이밍 신호 입력
5	PRAS'	하위 메모리 스트로브
6～13	RA0～RA7	다중화 어드레스 출력
14	R / W'	65C02 읽고 쓰기 제어신호
15	INH'	주기억장치 내의 전원을 +5V로 통제
16	OMA'	DMA 전송을 위한 제어 데이타버스
17	EN80'	보조 RAM 실행
18	KBD'	키보드 데이타 비트 0～6 실행
19	ROMEN2'	ROM을 ROMEN1'과 같게 실행
20	ROMEN1'	ROM을 ROMEN2'와 같게 실행
21	MD7	비트 7의 MMU플래그 내용
22	RW'245	74LS245 데이타버스 버퍼 제어
23	RAMEN'	메인 RAM을 실행
24	Cxxx	주변 카드의 메모리를 실행
25	+5V	전원
26～40	A15～A1	65C02 어드레스 입력

(3) PAL(Programmable Array Logic)

• PAL의 핀 번호도

14M	1	20	+5V
7M	2	19	PRAS'
3.58M	3	18	(n.c.)
H0	4	17	PCAS'
VID7	5	16	Q3
SEGB	6	15	φ0
GR	7	14	φ1
AMEN'	8	13	VID7M
80VID'	9	12	LDPS'
GND	10	11	ENTMG

• PAL 내의 각 핀들의 기능

핀번호	핀이름	기 능
1	14M	14.31818MHz 마스터 타이밍 신호
2	7M	7.15909MHz 타이밍 신호
3	3.58M	3.579545MHz 타이밍 신호
4	H0	수평 영상 타이밍 신호
5	VID7	비트 7의 영상 데이타
6	SEGB	영상 타이밍 신호
7	GR	그래픽 모드 실행
8	RAMEN'	RAM 실행
9	80VID'	80컬럼 모드 실행
10	GND	접지
11	ENTMG	마스터 타이밍 실행

핀번호	핀이름	기 능
12	LDPS'	쉬프트-레지스터 로드
13	VID7M	영상 도트 클럭 7~14MHz
14	φ1	Phase 1 시스템 클럭
15	φ0	Phase 0 시스템 클럭
16	Q3	타이밍 및 스트로브 신호
17	PCAS'	RAM 컬럼-어드레스 스트로브
18	N.C.	사용하지 않는 핀
19	PRAS'	RAM 하위 번지 스트로브
20	+5V	전원

§ 2. 하드웨어 사양

1. 문자 세트

키보드를 누를 때 발생하는 코드는 다음과 같다.

	Normal		Control		Shift		Both	
Key	Code	Char	Code	Char	Code	Char	Code	Char
DELETE	7F	DEL	7F	DEL	7F	DEL	7F	DEL
←	08	BS	08	BS	08	BS	08	BS
TAB	09	HT	09	HT	09	HT	09	HT
↓	0A	LF	0A	LF	0A	LF	0A	LF
↑	0B	VT	0B	VT	0B	VT	0B	VT
RETURN	0D	CR	0D	CR	0D	CR	0D	CR
→	15	NAK	15	NAK	15	NAK	15	NAK
ESC	1B	ESC	1B	ESC	1B	ESC	1B	ESC
SPACE	20	SP	20	SP	20	SP	20	SP
' "	27	'	27	'	22	"	22	"
, 〈	2C	,	2C	,	3C	〈	3C	〈
- _	2D	-	1F	US	5F	—	1F	US
. 〉	2E	.	2E	.	3E	〉	3E	〉
/ ?	2F	/	2F	/	3F	?	3F	?
0)	30	0	30	0	29	)	29	)
1 !	31	1	31	1	21	!	21	!
2 @	32	2	00	NUL	40	@	00	NUL
3 #	33	3	33	3	23	#	23	#
4 $	34	4	34	4	24	$	24	$

Key	Normal		Control		Shift		Both	
	Code	Char	Code	Char	Code	Char	Code	Char
5 %	35	5	35	5	25	%	25	%
6 ^	36	6	1E	RS	5E	^	1E	RS
7 &	37	7	37	7	26	&	26	&
8 *	38	8	38	8	2A	*	2A	*
9 (	39	9	39	9	28	(	28	(
; :	3B	;	3B	;	3A	:	3A	:
= +	3D	=	3D	=	2B	+	2B	+
[{	5B	[	1B	ESC	7B	{	1B	ESC
\ |	5C	\	1C	FS	7C	|	1C	FS
] }	5D	]	1D	GS	7D	}	1D	GS
` ~	60	`	60	`	7E	~	7E	~
A	61	a	01	SOH	41	A	01	SOH
B	62	b	02	STX	42	B	02	STX
C	63	c	03	ETX	43	C	03	ETX
D	64	d	04	EOT	44	D	04	EOT
E	65	e	05	ENQ	45	E	05	ENQ
F	66	f	06	ACK	46	F	06	ACK
G	67	g	07	BEL	47	G	07	BEL
H	68	h	08	BS	48	H	08	BS
I	69	i	09	HT	49	I	09	HT
J	6A	j	0A	LF	4A	J	0A	LF
K	6B	k	0B	VT	4B	K	0B	VT
L	6C	l	0C	FF	4C	L	0C	FF
M	6D	m	0D	CR	4D	M	0D	CR
N	6E	n	0E	SO	4E	N	0E	SO
O	6F	o	0F	SI	4F	O	0F	SI
P	70	p	10	DLE	50	P	10	DLE
Q	71	q	11	DC1	51	Q	11	DC1
R	72	r	12	DC2	52	R	12	DC2
S	73	s	13	DC3	53	S	13	DC3
T	74	t	14	DC4	54	T	14	DC4
U	75	u	15	NAK	55	U	15	NAK
V	76	v	16	SYN	56	V	16	SYN
W	77	w	17	ETB	57	W	17	ETB
X	78	x	18	CAN	58	X	18	CAN
Y	79	y	19	EM	59	Y	19	EM
Z	7A	z	1A	SUB	5A	Z	1A	SUB

다음 표는 Ⅱe의 40컬럼의 캐릭터와 확장 80컬럼(40 / 80컬럼)시의 캐릭터 표를 나타낸 것이다.

40컬럼			확장 80컬럼	
16진수	문자종류	형식	문자종류	형식
\$00~\$1F	대 문 자	Inverse	대 문 자	Inverse
\$20~\$3F	특수문자	Inverse	특수문자	Inverse
\$40~\$5F	대 문 자	Flash	마우텍스트	
\$60~\$7F	특수문자	Flash	소 문 자	Inverse
\$80~\$9F	대 문 자	Normal	대 문 자	Normal
\$A0~\$BF	특수문자	Normal	특수문자	Normal
\$C0~\$DF	대 문 자	Normal	대 문 자	Normal
\$E0~\$FF	소 문 자	Normal	소 문 자	Normal

2. 화면 디스플레이 문자 맵

다음부터 소개하는 그림들은 애플Ⅱe의 화면 디스플레이시·각 화면에 해당하는 그림들이다.

• 40컬럼(표준 화면) 텍스트 디스플레이 화면 맵(40×24문자)

			\$00 0	\$01 1	\$02 2	\$03 3	\$04 4	\$05 5	\$06 6	\$07 7	\$08 8	\$09 9	\$0A 10	-------	\$1F 31	\$20 32	\$21 33	\$22 34	\$23 35	\$24 36	\$25 37	\$26 38	\$27 39
0	\$400	1024																					
1	\$480	1152																					
2	\$500	1280																					
3	\$580	1408																					
4	\$600	1536																					
5	\$680	1664																					
6	\$700	1792																					
7	\$780	1920																					
8	\$428	1064																					
9	\$4A8	1192																					
10	\$528	1320																					
11	\$5A8	1448																					
12	\$628	1576																					
13	\$6A8	1704																					
14	\$728	1832																					
15	\$7A8	1960																					
16	\$450	1104																					
17	\$4D0	1232																					
18	\$550	1360																					
19	\$5D0	1488																					
20	\$650	1616																					
21	\$6D0	1744																					
22	\$750	1872																					
23	\$7D0	2000																					

3. ESC코드 및 기타 키 입력

(1) ESC 코드

ESC-@

Window를 클리어시키고 커서를 Home위치로 하고 Escape 모드에서 빠져나온다.

ESC-A or ESC-a

커서를 오른쪽으로 한 칸 이동시킨 후 Escape 모드에서 빠져나온다.

ESC-B or ESC-b

커서를 왼쪽으로 한 칸 이동시킨 후 Escape 모드에서 빠져나온다.

ESC－C or ESC-c

커서를 아래로 한 칸 이동시킨 후 Escape 모드에서 빠져나온다.

ESC-D or ESC-d

커서를 위로 한 칸 이동시킨 후 Escape 모드에서 빠져나온다.

ESC-E or ESC-e

줄 끝까지 지우고 Escape 모드에서 빠져나온다.

ESC-F or ESC-f

Window의 맨 밑까지 지우고 Escape 모드에서 빠져나온다.

ESC-I or ESC-i or ESC-↑

커서를 위로 한 칸 이동시키고 Escape 모드에 남아있는다.

ESC-J or ESC-j or ESC-←

커서를 왼쪽으로 한 칸 이동시키고 Escape 모드에 남아있는다.

ESC-K or ESC-k or ESC-→

커서를 오른쪽으로 한 칸 이동시키고 Escape 모드에 남아있는다.

ESC-M or ESC-m or ESC-↓

커서를 아래로 한 칸 이동시키고 Escape 모드에 남아있는다.

ESC-4

80컬럼 펌웨어가 작동할 때 40컬럼 모드로 바꾼다.

ESC-8

80컬럼 펌웨어가 작동할 때 80컬럼 모드로 절환된다.

위 두 개는 80컬럼 펌웨어 작동시 40/80컬럼 절환 스위치 역할을 한다.

ESC-CTRL-D

캐리지 리턴, 라인피드, 벨소리, Backspace를 제외한 CTRL 캐릭터는 효과가 없어진다.

VLINE($F828)

저해상 그래픽의 수직 블럭선을 그린다.

2. 80컬럼 사용시의 키 입력

다음 표를 보면 80컬럼 사용시에 입력한 특수키들의 기능을 잘 알 수 있을 것이다.

• 80컬럼 펌웨어가 OFF되었을 때 CTRL 캐릭터

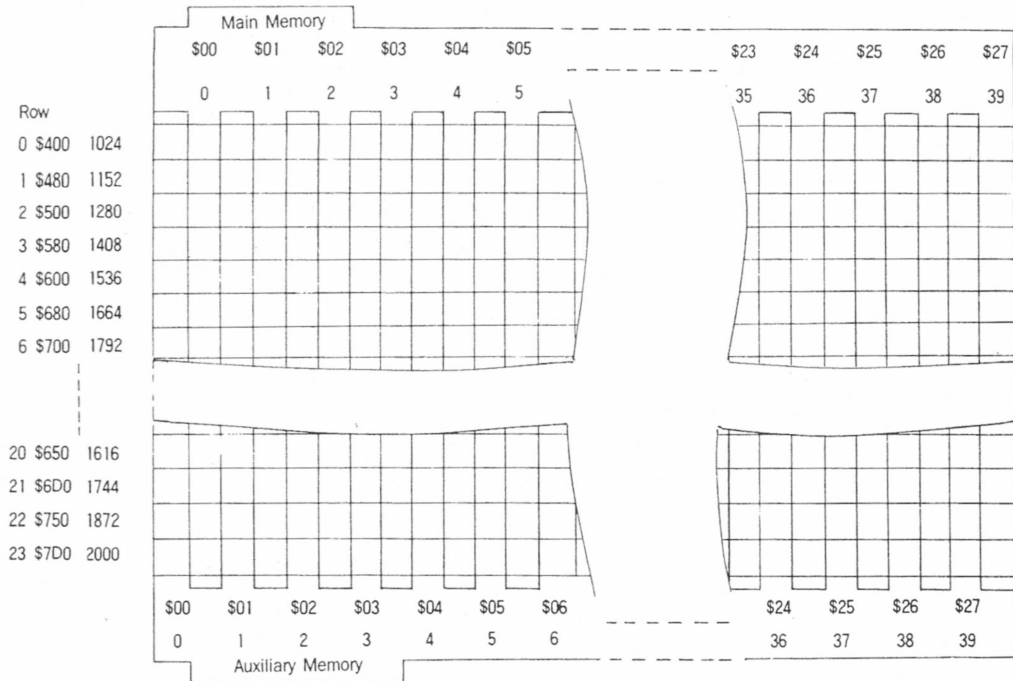
입력	아스키 이름	설 명
CTRL-G	BEL	1000Hz의 톤을 0.1초 동안 발생시킨다.
CTRL-H	BS	커서를 Backspace시킨다.
CTRL-J	LF	현재 커서의 위치에서 라인피드를 일으킨다.
CTRL-M	CR	리턴키를 누른 것과 같다.

• 80컬럼 펌웨어가 ON되었을 때 CTRL 캐릭터

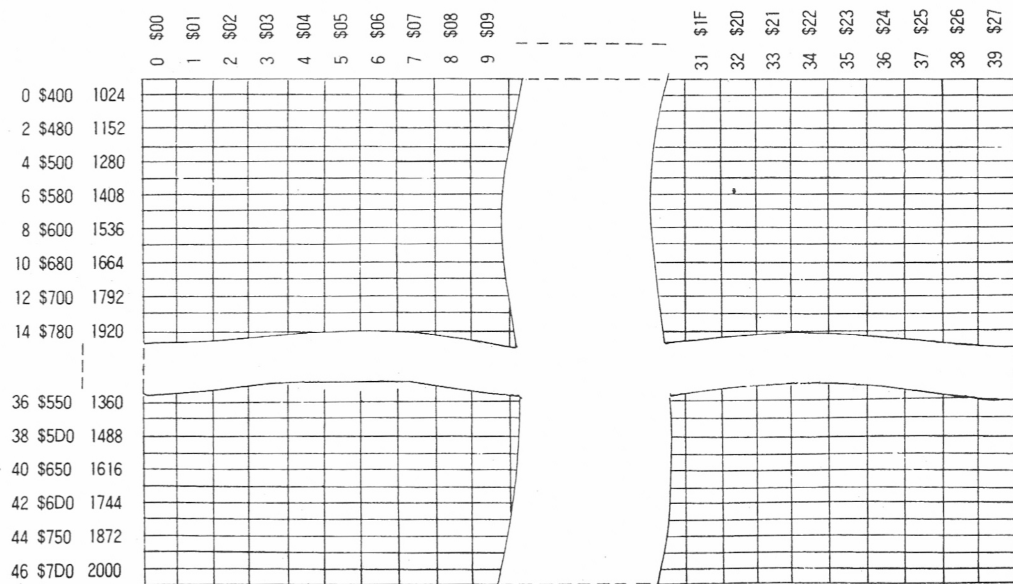
입력	아스키 이름	설 명
CTRL-G	BEL	OFF때와 동일
CTRL-H	BS	OFF때와 동일
CTRL-J	LF	OFF때와 동일
⊕CTRL-K	VT	현재의 커서 위치부터 화면 끝까지 지운다.
⊕CTRL-L	FF	HOME 명령과 동일하다.
CTRL-M	CR	OFF때와 동일
⊕CTRL-N	SO	Normal을 취한다.
⊕CTRL-O	SI	Inverse를 취한다.
⊕CTRL-Q	DC1	40컬럼 모드로 세트한다.
⊕CTRL-R	DC2	80컬럼 모드로 세트한다.
CTRL-S	DC3	리스트시 잠깐 멈추게 한다. 아무 키나 누르면 다시 된다.
⊕CTRL-U	NAK	80컬럼 비디오 펌웨어를 해제한다.
⊕CTRL-V	SYN	커서의 현재 위치를 한 줄 스크롤다운시킨다.
⊕CTRL-W	ETB	커서의 현재 위치를 한 줄 스크롤업시킨다.
CTRL-X	CAN	마우스 텍스트 표시를 멈춘다.
⊕CTRL-Y	EM	화면을 지우지 않고 현재 Window 상단 왼쪽 코너에 커서를 위치시킨다.
⊕CTRL-Z	SUB	커서가 위치한 줄을 지운다.
CTRL-[	ESC	마우스 텍스트 표시 가동
⊕CTRL-\	FS	스페이스 한 개를 발생시키고 커서의 위치를 오른쪽으로 한 칸 옮긴다.
⊕CTRL-]	GS	커서가 현재 위치한 곳부터 그 줄 끝까지 지운다(window 범위 안에서).
CTRL-_	US	스크롤없이 커서를 한 칸 위로 옮긴다.

※ ⊕표시는 직접 입력 모드에서는 할 수 없고 프로그램상에서만 가능하다.

• 80컬럼 텍스트 디스플레이 화면 맵(80×24문자)



• 저해상도 그래픽 디스플레이 맵(40×40도트+4라인 텍스트)



기능	10진 번지	16진 번지	ACCESS
아날로그 Input 2	49254 or －16282	$C066	읽기만 함
아날로그 Input 3	49255 or －16281	$C067	읽기만 함

§3. 모니터 펌웨어 및 키입력

애플Ⅱe의 모니터 롬에는 여러가지 서브루틴이 존재하게 되는데 이것들을 잘 사용하면 프로그램의 능률을 올릴 수 있다.

1. 주요 펌웨어들

아래에 열거한 번지들은 위 모니터 루틴의 시작 번지를 의미한다.

BASICIN($C305)

80컬럼이 작동할 때 사용되는 루틴으로 체커 모양의 커서를 표시하고 키보드로부터 키보드 입력을 받는다.

BASICOUT($C307)

이것 역시 80컬럼이 작동할 때 사용되는 루틴으로 문자를 화면에 표시한다.

CLREOL($FC9C)

커서가 현재 위치한 곳부터 그 줄 끝까지 지운다.

CLEOLZ($FC9E)

커서의 위치를 Y레지스터 내용에 담아 그 줄 끝까지 지운다.

CLREOP($FC42)

Window의 아래까지 지운다.

CLRSCR($F832)

저해상 그래픽 화면을 지운다.

CLRTOP($F836)

저해상 그래픽 화면의 세로로 40줄을 지운다.

COUT($FDED)

CSW를 저장하고 Output루틴을 부른다.

COUT1($FDF0)

문자를 화면에 표시한다.

CROUT($FD8E)

캐리지 리턴을 발생시킨다.

CROUT1($FD8B)

줄 끝까지를 지우고 캐리지 리턴을 발생시킨다.

GETLN($FD6A)

프롬프트를 표시하고 RDKEY로부터 문자 입력을 받아들인다.

HLINE($F819)

저해상 그래픽의 수평 블럭선을 그린다.

HOME($FC58)

화면을 전부 지우고 커서를 상단 왼쪽 코너에(window 설정시는 그에 해당하는 코너) 위치시킨다.

KEYIN($FD1B)

80컬럼이 작동하지 않을 때, 즉 40컬럼 모드에서 체커보드 모양의 커서를 표시하고 키입력을 받는다.

PLOT($F800)

저해상도 화면에 한 블럭을 그린다.

PRBL2($F94A)

256개의 스페이스를 출력한다.

PRBYTE($FDDA)

HexaDecimal 바이트를 인쇄한다.

PRERR($FF2D)

ERR표시를 벨(CTRL-G)소리와 함께 출력한다.

PRHEX($FDE3)

4bit를 16진 숫자로 인쇄한다.

PRNTAX($F941)

A, X레지스터의 내용을 16진수로 인쇄한다.

RDKEY($FD0C)

깜빡이는 커서를 표시하고 키입력을 기다리는 Input 루틴이다.

SCRN($F871)

저해상 그래픽의 블럭 색깔을 읽는다.

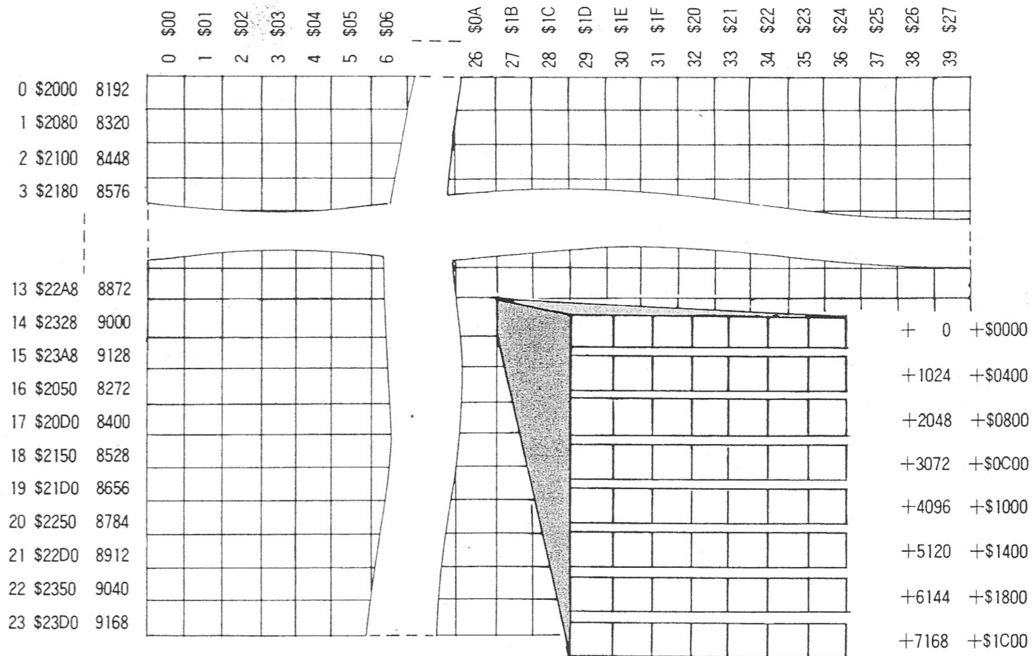
SETCOL($F864)

저해상 그래픽의 블럭의 색깔을 세트한다.

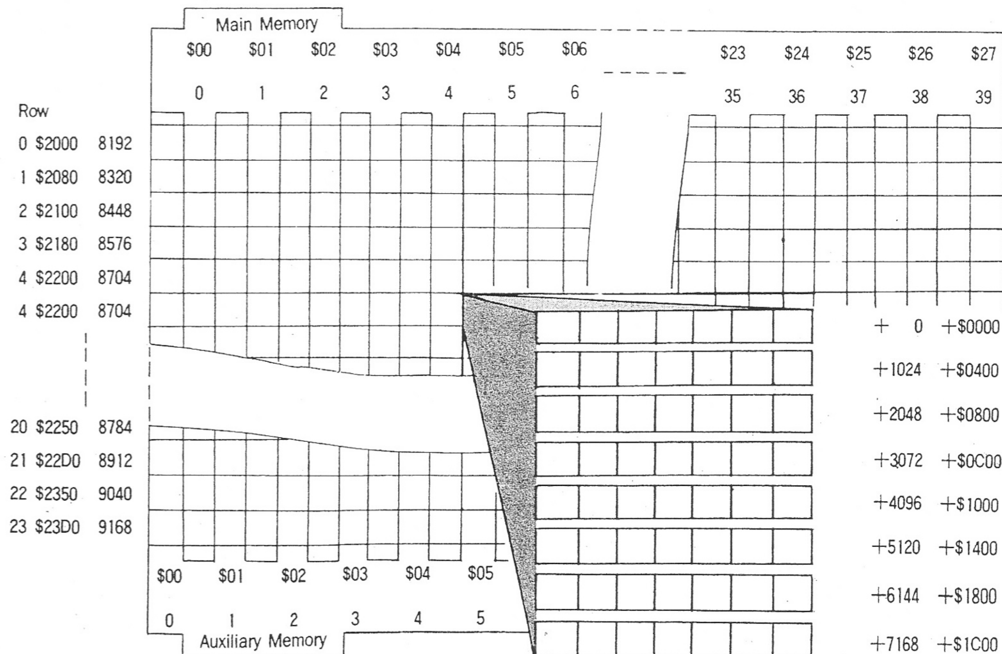
VTABZ($FC24)

커서의 세로 위치를 세트한다.

- 고해상도 그래픽 디스플레이 맵(280×192도트)



- 더블 고해상도 그래픽 디스플레이 맵(560×192 도트)



3. 카세트 및 게임 I/O

(1) 카세트 Input, Output

IIe본체 뒤에 보면 Input, Output이라고 써있는 두 개의 잭을 볼 수 있을 것이다. 이것은 카세트 레코더로 데이타를 입출력할 때 카세트 레코더와 IIe본체를 연결하는 것으로, 각각 한 개의 소프트 스위치를 갖고 있다. 그리고 소프트 스위치로 억세스함으로써 0~25mV 또는 25mV~0까지의 신호를 잭을 통해 내보낼 수 있다.

Input 소프트 스위치 $C060(49248)
Output 소프트 스위치 $C020 (49184)

(2) 보조 I/O 메모리 위치

아래 표에서 어눈시에이터(annunciator) 출력은 번지를 억세스하면 게임I/O에 해당하는 핀이 ON/OFF된다. 이 어눈시에이터는 표에 있는 번지로 ON/OFF되는데, 현재 어떤 상태인가를 프로그램으로 알 수 있는 방법은 없다. 단지 리셋 사이클(reset cycle) 중 어눈시에이터 0, 1을 OFF로 하고 2, 3을 ON으로 한다.

- 게임 I/O 및 카세트 소프트 스위치 일견표

기능	10진 번지	16진 번지	ACCESS
Speaker	49200 or −16336	$C030	읽기만 함
Cassette Out	49184 or −16352	$C020	읽기만 함
Cassette In	49248 or −16288	$C060	읽기만 함
어눈시에이터 0 ON	49241 or −16295	$C059	·
어눈시에이터 0 OFF	49240 or −16296	$C058	·
어눈시에이터 1 ON	49243 or −16293	$C05B	·
어눈시에이터 1 OFF	49242 or −16294	$C05A	·
어눈시에이터 2 ON	49245 or −16291	$C05D	·
어눈시에이터 2 OFF	49244 or −16292	$C05C	·
어눈시에이터 3 ON	49247 or −16289	$C05F	·
어눈시에이터 3 OFF	49246 or −16290	$C05E	·
Strobe Output	49216 or −16320	$C040	읽기만 함
스위치 Input 0(open Apple)	49249 or −16287	$C061	읽기만 함
스위치 Input 1(solid Apple)	49250 or −16286	$C062	읽기만 함
스위치 Input2	49251 or −16285	$C063	읽기만 함
아날로그 Input Reset	49264 or −16272	$C070	
아날로그 Input 0	49252 or −16284	$C064	읽기만 함
아날로그 Input 1	49253 or −16283	$C065	읽기만 함

ESC-CTRL-E

위에서 효과가 없어진 것을 다시 원상복구한다.

ESC-CTRL-Q

80컬럼 펌웨어가 작동할 때 80컬럼 펌웨어에서 완전히 빠져나온다.

(2) 펌웨어 프로토콜 기능

다음 표는 슬롯 3번의 펌웨어 프로토콜 기능을 나타내었다.

번지	수 치	설 명
$C30B	$01	펌웨어 카드의 일반적인 기호 바이트
$C30C	$88	80컬럼 카드 장치 기호
$C30D	$ii	$C3ii는 초기화 루틴의 엔트리포인트(PINIT)
$C30E	$rr	$C3rr은 Read 루틴의 엔트리포인트(PREAD)
$C30F	$ww	$C3ww는 Write 루틴의 엔트리포인트(PWRITE)
$C310	$ss	$C3ss는 Status 루틴의 엔트리포인트(PSTATUS)

(3) Pascal 비디오 컨트롤 기능

키입력은 CTRL키와 동시에 누르는 것을 원칙으로 한다.

키입력	Hex	기 능
CTRL-E or e	$ 05	커서를 켠다.
CTRL-F or f	$ 06	커서를 끈다.
CTRL-G or g	$ 07	Bell 소리를 낸다.
CTRL-H or h	$ 08	커서를 왼쪽으로 1컬럼 이동시킨다.
CTRL-J or j	$ 0A	커서를 아래로 1 Row 이동시킨다.
CTRL-K or k	$ 0B	화면 끝까지 지운다.
CTRL-L or l	$ 0C	화면을 지우고 커서를 화면의 상단 왼쪽 코너에 위치시킨다.
CTRL-M or m	$ 0D	커서를 컬럼0으로 이동시킨다.
CTRL-N or n	$ 0E	Subsequent 문자를 Normal로 표시
CTRL-O or o	$ 0F	Subsequent 문자를 Inverse로 표시
CTRL-V or v	$ 16	한 줄 스크롤업시킨다.
CTRL-W or w	$ 17	한 줄 스크롤다운시킨다.
CTRL-Y or y	$ 1P	커서를 Home 위치로 이동시킨다.
CTRL-Z or z	$ 1A	커서가 위치한 줄 전체를 지운다.
CTRL-| or \	$ 1C	커서를 오른쪽으로 1컬럼 이동시킨다.
CTRL-} or]	$ 1D	커서가 위치한 곳부터 그 줄 끝가지 지운다.
CTRL-^ or 6	$ 1E	GOTOxy 시퀀스이다.
CTRL-_	$ 1F	커서를 한 줄 위로 이동시킨다.

4. 제로페이지의 사용

(1) 모니터가 사용하는 제로페이지 영역

High Nibble of Address	Low Nibble of Address $0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00																
$10																•
$20	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$30	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$40	•	•	•	•	•	•	•	•	•	•					•	•
$50	•	•	•	•	•	•										
$60																
$70																
$80																
$90																
$A0																
$B0																
$C0																
$D0																
$E0																
$F0																

(2) 애플소프트가 사용하는 제로페이지 영역

High Nibble of Address	Low Nibble of Address $0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00	•	•	•	•	•	•					•	•	•	•	•	•
$10	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
$20							•	•					•	•		•
$30	•		•	•									•	•	•	•
$40																
$50	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$60	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$70	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$80	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$90	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$A0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$B0	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
$C0	•	•	•	•	•	•	•	•	•	•	•	•	•	•		
$D0	•	•	•	•	•	•	•		•	•	•	•	•	•	•	•
$E0	•	•	•	•	•	•	•	•	•	•	•					
$F0	•	•	•	•	•	•	•	•	•	•						•

(3) 정수 베이직에서 사용하는 제로페이지 영역

High Nibble of Address	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00														●		
$10																
$20																
$30																
$40											●	●	●	●		
$50						●	●	●	●	●	●	●	●	●	●	●
$60	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$70	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$80	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$90	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$A0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$B0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$C0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$D0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
$E0																
$F0															●	●

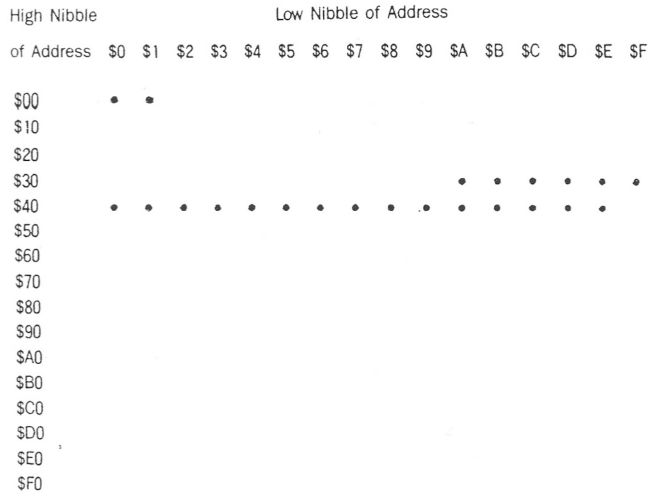
(Column header group: Low Nibble of Address)

(4) DOS 3.3에서 사용하는 제로페이지 영역

High Nibble of Address	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	$F
$00																
$10																
$20							●	●			●	●	●	●	●	●
$30						●	●	●	●	●					●	●
$40	●	●	●	●	●	●	●	●	●		●	●	●	●		
$50																
$60								●	●	●	●					●
$70	●															
$80																
$90																
$A0																●
$B0	●															
$C0											●	●	●	●		
$D0									●							
$E0																
$F0																

(Column header group: Low Nibble of Address)

(5) ProDOS MLI와 Disk-Driver가 사용하는 제로페이지 영역



5. Page 3의 Vector

Vector 번지	기 능
$3F0, $3F1	BRK가 걸렸을 때 서브루틴 번지, Default $59, $FA가 들어있다.
$3F2, $3F3	리셋 벡터
$3F4	Power-up Byte
$3F5, $3F6, $3F7	애플소프트에서 &마크가 걸렸을 때 점프 번지, Default $4C, $58, $FF가 들어있다.
$3F8, $3F9, $3FA	모니터모드에서 CTRL-Y가 걸렸을 때 점프 번지
$3FB, $3FC, $3FD	Non-maskable 인터럽트가 걸렸을 때 점프 번지
$3FE, $3FF	인터럽트 벡터

§4. 미니 어셈블러

Ⅱe에서 새로이 생겨난 것은 바로 이 미니 어셈블러이다. 이것은 기존의 정수 베이직에서도 있었으나, 애플소프트에서는 삭제되었었다. 이것은 레이블(label)을 사용할 수 없다는 단점은 있으나, 나름대로 편히 쓸 수 있다.

1. 미니 어셈블러의 어드레스 형식

어드레싱 모드	형 식	어드레싱 모드	형 식
Accumulator		Indexed zero page	${address}, X
Implied			${address}, Y
Immediate	#${value}	Indexed absolute	${address}, X
Absolute	${address}		${address}, Y
Zero Page	${address}	Relative	${address}
		Indexed indirect	(${address}, X)
		Indirect indexed	(${address}), Y
		Absolute indirect	(${address})

• 참고 : 미니 어셈블러의 작동

작동 : 오리지널][e에서는 모니터 모드에서 F666G로 작동시키고 Enhanced][e에서는 모니터 모드에서 !를 타이프하고 리턴키를 누르면 된다.

중지 : 오리지널][e에서는 $FF69G로 중지시키고 Enhanced][e에서는 리턴키로 중지시킨다.

2. BRK를 다루는 자료

명 칭	위 치
Program Counter(low)	$3A
Program Counter(high)	$3B
Encoded Memory State	$44
Accumulator	$45
X레지스터	$46
Y레지스터	$47
Status 레지스터	$48

제 3 장

CPU 및 기계어

§1. 일반적인 CPU 개관
§2. 애플의 CPU인 6502에 대하여
§3. 6502의 표준 CPU, 65C02
§4. 애플 IIe의 한계를 넘는다

§ 1. 일반적인 CPU 개관

애플 Ⅱe라고 해서 CPU를 다른 것을 쓰는 것은 아니다. 특히 국내에 보급된 Ⅱe 기종은 기존의 애플 Ⅱ+와 거의 같은 CPU를 사용하고 있다. 그러나 65C02의 경우는 사정이 좀 다르기 때문에 뒤에서 서술하기로 하겠다.

본론에 들어가기 전에 먼저 CPU에 대해 간단히 알아보자 이 책을 보는 독자들 가운데 CPU에 대해 모르는 사람은 없겠지만, CPU의 계통에 대해서는 자세히 알려진 것이 없어 간략히 설명하기로 한다.

현재 CPU에는 2대 계통이 있다. 먼저 1971년에 발표한 인텔 4004를 바탕으로 하여 8008, 8080, Z-80으로 발전해 나간 80계(系)와 모토롤라 6800을 시작으로 하는 60계가 그것이다. 양대 CPU가 다 나름대로의 특성이 있으나, 그렇다고 해서 완전히 다른 것은 아니다. 표 3. 1은 80계와 60계의 차이점을 나타낸 것이다.

80 계	60 계
전자계산기의 성능을 고도화하는 목적에서 개발된 것.	중 · 대형 컴퓨터를 소형화하는 목적에서 개발된 것.
내부 레지스터가 잘 정비되어 있음.	내부 레지스터는 80계에 비해 뒤떨어짐.
명령 체계는 별로 잘 되어있지 않음.	스마트한 명령 계통으로 다루기 편함.

표 3. 1 60계와 80계의 차이점

우리가 사용하는 애플 Ⅱe도 60계 CPU인 6502를 사용하고 있다. 따라서 어드레싱 모드 등의 명령 계통은 비교적 잘 정비되어 있으나, 80계 CPU인 Z-80A에 비해 내부 레지스터의 수가 엄청나게 적은 것을 알 수 있다. 그러나 60계 CPU가 대형 컴퓨터를 축소하는 데에서 생긴 것이므로 애플 Ⅱe는 '대형 컴퓨터의 축소판'이라고 말해도 무방할 것이다.

§ 2. 애플의 CPU인 6502에 대하여

1. 6502의 특징

6502는 모토롤라에서 퇴직한 직원들이 모스테크놀로지(MOS Technology)라는 회사를 세워 그 곳에서 만든 CPU이다. 연산 속도를 중시한 간단한 CPU라고 하는데, 그렇기 때문에 다른 CPU에 비해 명령어 수가 부족하다.

사실 애플사가 왜 수많은 CPU들 가운데서 유독 6502를 채택했는지는 아직 정설이 없다. 아마 자금 부족으로 당시에 구할 수 있는 CPU들 중에서 가장 값이 싼 6502를 골랐다는 것을 정설로 받아들여야

할 듯하다. 중고 폭스바겐을 팔아 마련한 단돈 1천2백 달러를 가지고 세운 회사이므로 자금은 이 회사의 적잖은 문제였을 것이다(애플의 그래픽 화면이 메모리와 연속적으로 연결되어 있지 않고 세 부분으로 끊어져 있는 것은 두 개의 OR 게이트 부품을 줄이기 위함이었다고 한다).

6502의 장점은 다음과 같다.

① 처리 속도가 빠르다. Z-80A가 3.6MHz로 동작하는 데 비해 6502는 1MHz로 동작하는데도 거의 속도차를 느끼지 않는다. 명령어 수가 적은 것이 이럴 때는 장점이 된다.

② 6502의 정평있는 파이프라이닝(pipelining) 처리는 6502의 수행 속도를 한층 높여준다.

③ 어드레싱 모드 체계 등이 다른 것에 비해 잘 되어 있다.

④ 명령어 중 제로페이지(zero page)에 관한 것은 6502만의 독특한 점으로, 처리 속도의 향상과 처리 능률의 증가를 가져온다.

이런 이유 때문에 6502는 애플뿐만 아니라 코모도어(Commodore)의 VIC나 CBM4032 등의 기종에도 사용되어 그 위력을 나타내고 있다.

2. 6502의 내부 레지스터

그림 3. 1은 6502의 내부 레지스터를 간단히 나타낸 것이다.

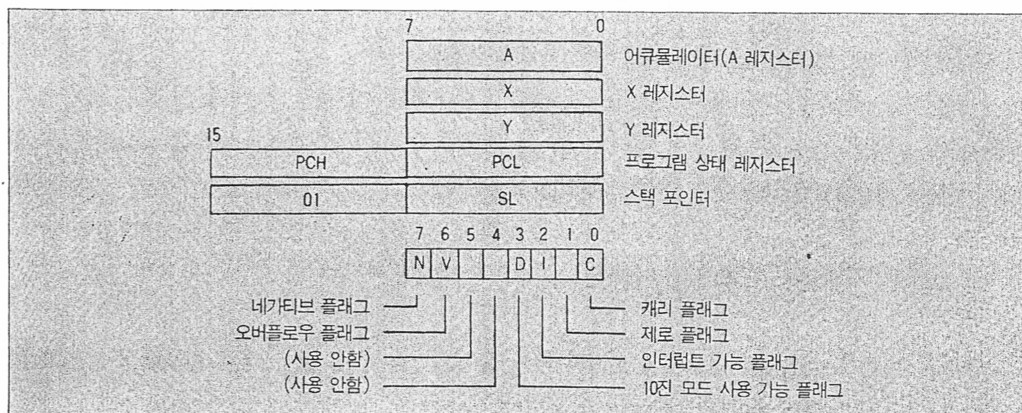


그림 3. 1 6502의 내부 레지스터

가장 위의 A라고 표시된 것이 어큐뮬레이터(accumulator 누산기)라고 불리는 것으로, 대부분의 연산이 이 곳에서 이루어진다. 그 밑에 X, Y라고 된 것은 두 개의 인덱스 레지스터이다. 일시로 데이타를 저장해 넣는 공간으로 쓰기도 하지만, 주로 간접 어드레싱 모드에서 인덱스용으로 쓴다. 그 밑에 PCH, PCL이라고 되어있는 것은 프로그램 상태 레지스터(program status register)로 현재 실행되는

곳의 어드레스를 담는다. 다음 것은 스택 포인터인데, 상위 바이트는 $01로 고정이고 하위 바이트가 SL에 들어간다.

그 밑의 것은 한 비트가 각각 중요한 의미를 가진다. 왼쪽부터 N, V, D, I, Z, C 플래그라고 하는데, 각각의 의미는 다음과 같다.

N 플래그—네가티브 플래그라고 하며, 어떤 연산을 하여 그 결과가 마이너스(−)가 되었을 때는 이 플래그는 1, 그밖에는 0이다.

V 플래그—오버플로우 플래그라고 한다. 8비트로는 0~255까지의 수를 사용할 수 있는데, 최상위 비트(MSB)를 부호 비트로 하면 8비트는 −128~+127의 값을 가진다. 이 때 127은 2진수로 01111111인데, 여기서 1을 더하면 10000000이 되고 숫자로는 −128이 된다. 이럴 때 이것을 오버플로우(overflow)라고 하고, V 플래그가 1로 세트된다.

D 플래그—10진 플래그라고 한다. 이 플래그가 세트된 경우에는 10진 BCD(Binary Coded Decimal) 모드가 되어 BCD 연산을 하게 된다. 클리어되었을 때에는 보통의 2진 연산 모드로 된다.

I 플래그—인터럽트 가능/불가능 플래그라고 한다. 이것이 세트되었을 때에는 인터럽트가 가능하게 되고, 클리어되었을 때에는 인터럽트가 불가능하다.

Z 플래그—어떤 명령을 실행하여 그 결과가 0이 되었을 때 이 플래그는 1로 세트되고, 그밖에는 0이다.

C 플래그—캐리 플래그라고 한다. 캐리란 인터럽트와는 달리 MSB를 부호 비트로 하지 않는 경우에 발생한다. 현재의 비트값이 11111111일 때, 여기에 1을 더하면 값이 100000000이 되어 8비트를 넘게 된다. 이 때 가장 왼쪽의 1은 무시되고 캐리 플래그가 세트된다.

지금까지는 6502의 CPU 레지스터 구조에 대해 알아보았다. 그림 3. 1에서 보는 것과 같이 내부 레지스터의 수가 적음을 알 수 있다. 하지만 이 점 때문에 이 CPU를 기능이 떨어진다고 말할 수 없는 것은 현재까지 애플용으로 나온 수많은 CPU가 보증한다.

3. 6502의 어드레싱 모드와 사용예

그러면 이번에는 6502의 어드레싱 모드에 대해 알아보기로 하자. 총 13개의 어드레싱 모드가 6502에서 쓰이는 데 이것을 정리해 보면 다음과 같다.

임플라이드 어드레싱 모드(Implied)

1바이트의 명령으로 어떤 조작을 한다.

예 CLD, INY, DEX, SEC 등

어큐뮬레이터 어드레싱 모드(Accumulator)

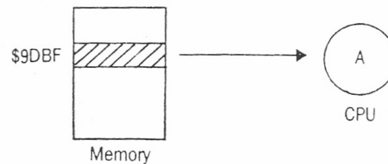
1바이트의 명령으로 어큐뮬레이터를 조작한다.

예 ASL, LSR, ROL, ROR 등

절대 어드레싱 모드(Absolute)

3바이트의 명령으로 첫 1바이트가 명령어이고 뒤의 2, 3바이트에서 실효 어드레스를 보여준다.

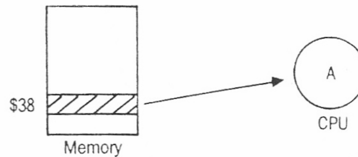
예 LDA $9DBF



제로페이지 어드레싱 모드(Zero page)

2바이트의 명령으로 절대 어드레싱과 비교해서 실효 어드레스가 1바이트로 줄어든 것.

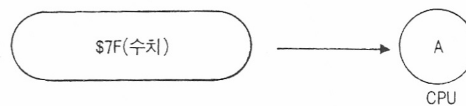
예 LDA $38



직접 어드레싱 모드(Immediate)

2바이트의 명령으로, 2바이트째가 바로 데이타가 된다.

예 LDA #$7F



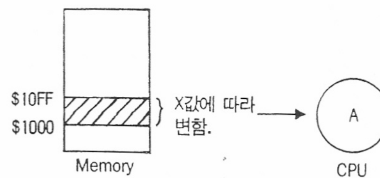
인덱스 절대 어드레싱 모드(IND, X)

3바이트의 명령으로 2, 3바이트째의 값에 X 또는 Y 레지스터의 값을 더한 것이 실효 어드레스가 된다.

예 LDA $1000, X (X=$20)

$1000+X=$1020

즉 LDA $1020과 같아진다.



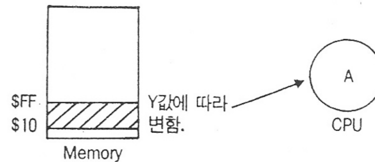
인덱스 제로페이지 어드레싱 모드(Zero, X)

2바이트의 명령으로 2바이트째의 값에 X 또는 Y 레지스터의 값을 더한 것이 실효 어드레스가 된다.

예 LDA $10, Y(Y=$2A)

$10+Y=$3A

주 만일 LDA $10, Y(Y=$FA)일 때, $10+Y=$10A이지만 결과는 $0A가 된다(제로페이지이므로).



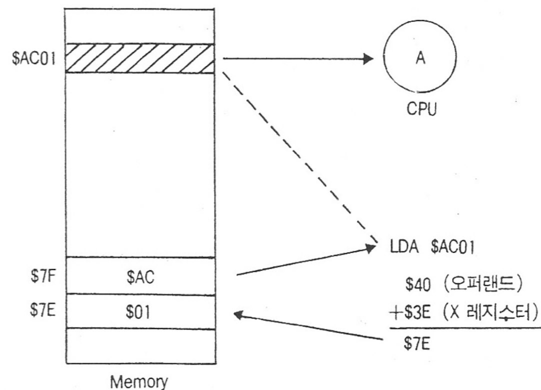
인덱스 간접 어드레싱 모드((Indirect, X))

2바이트의 명령으로 2바이트째의 값에 인덱스 레지스터의 X 레지스터의 값을 더한 것이 실효 어드레스의 하위 바이트를 취하고 그 다음 어드레스에서 실효 어드레스의 상위 바이트를 취한다.

예 LDA ($40, X) (X=$3E)

① 괄호 안을 더한다.→LDA ($7E)

② $7E, $7F의 내용을 어드레스로 한다. 결국 LDA $AC01과 같다.



간접 인덱스 어드레싱 ((Indirect), Y)

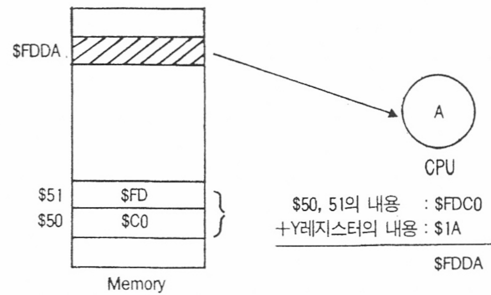
2바이트의 명령으로 2바이트째가 가리키는 어드레스와 다음의 어드레스의 내용에 인덱스 레지스터의 Y 레지스터의 값을 더한 것이 실효 어드레스가 된다.

예 LDA ($50), Y (Y=$1A)

① $50, $51번지의 값을 취한다. ($FDC0)

② 여기에 Y레지스터의 값을 더한다.($FDDA)

결국 LDA $FDDA와 같다.



렐러티브 어드레싱 (Relative)

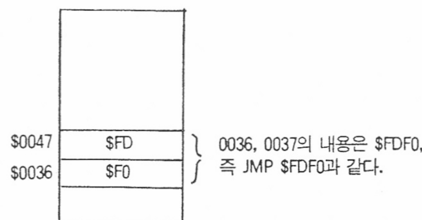
2바이트의 명령으로 브랜치(branch) 명령에 사용된다. 브랜치선의 어드레스는 〈그 명령의 어드레스〉 +2+〈2바이트째의 수치〉로 된다. 수치는 −128에서 127 사이의 값을 가진다.

이 브랜치선의 어드레스를 구하는 것은 쉽지 않으므로 보통 어셈블러를 사용한다.

간접 어드레싱 (Indirect)

3바이트의 명령으로 오직 JMP 하나이다. 2바이트, 3바이트째에서 보여주는 어드레스의 내용이 점프할 자리가 된다.

예 JMP ($0036)



4. 6502A, 6502B, 6502C는 무엇인가

다음에 설명할 65C02를 이해하기 위해서는 먼저 6502에 대한 확실한 이해가 있어야 하기 때문에, 어드레싱 모드와 내부 레지스터에 대해 조사해 보았다.

그러면 6502A는 무엇인가? 현재 우리나라에서 발매되는 애플Ⅱe에는 이 CPU가 탑재되어 있다. 클럭 속도도 2MHz로 기존 6502의 두 배라고 한다. 그렇다면 기존 Ⅱ+의 두 배의 속도로 동작한다는 말인데, 실은 그렇지 않다. 그 이유는 무엇일까?

대답은 간단하다. 6502A는 2MHz로 '동작 가능한' CPU이지만 국산 Ⅱe에서는 1MHz로 동작한다. 그렇기 때문에 속도가 같은 것이다. 그러면 이를 2MHz로 동작시켜 주면 될 것이라고 생각할 수도 있다. 클럭 속도만 두 배 증가해 주면 그렇게 될 듯도 하다. 하지만 그렇게 간단한 것이라면 메이커에서

2MHz로 동작하도록 제품을 내놓았을 것이다.

사실은 이렇다. CPU만 2MHz가 되었다고 모든 것이 OK는 아니다. 램의 억세스 속도 또한 그에 따라주어야 한다. 현재 판매되는 Ⅱe에는 150ns(나노초(秒))의 RAM이 들어있는데, 이것을 교체해야만 한다. 또 클럭 속도를 올리면 자연히 비디오 회로의 동기 주파수가 변동하는데, 이것은 15.75KHz를 그대로 유지해야 한다. 따라서 2MHz로 동작시키려면 여러가지 부수 회로가 붙어야 한다. 시중에는 이들 회로를 모두 포함하는 인터페이스 카드가 판매되고 있는데, 이것을 사서 끼우면 그제서야 애플은 두 배, 세 배의 속도로 움직이게 된다. 참고로 6502B는 3MHz CPU이다.

그러면 이제 구식 6502 이야기는 그만하고 65C02로 들어가기로 하자.

§3. 애플Ⅱe의 표준 CPU, 65C02

65C02는 미국에서는 이미 Ⅱc, Ⅱe Enhanced 등에 쓰이고 있고 지금은 6502를 거의 압도한 상태이다. 게다가 65C02 전용의 인스트럭션을 쓴 소프트웨어도 나오는 것을 볼 때 6502가 Ⅱe에 표준으로 자리잡아 가고 있다. 따라서 우리나라에서도 6502 정품을 쓰는 Ⅱe를 생산하는 업체가 몇 있지만 아직은 부족한 실정이다. 많은 업체에서 생산되길 기대한다.

1. 6502의 단점

앞 장에서 6502의 장점과 특징에 대해 이야기했는데, 이제는 6502의 단점에 대해 몇마디 해보겠다. 솔직히 말하면 6502에는 문제점(?)이 있다는 것이다. 기계어를 많이 써보지 않은 독자는 이러한 문제를 접해본 적이 없을 것이다. 그러나 한번 실험해 보라.

(1) 기계 자체의 문제

① 명령어의 수와 어드레싱 모드가 부족하다. 앞에서 60개의 장점이 스마트한 명령 체계라고 했는데, 체계는 잘 되어 있으나 그 갯수는 부족하다. 그 때문에 배우기 쉽기는 하지만 한 번에 할 수 있는 명령은 2단계 이상을 써서 나타내야 하는 경우가 많다.

② 순간의 실수에 의한 말썽이 있다. 실제로 6502는 사용상의 에러에 의한 문제가 적지 않으며, 애플컴퓨터가 말썽을 잘 일으키는 이유 중의 하나도 바로 그 때문이라고 할 수 있다. 가령 실제로 존재하지 않는 명령어를 사용했을 경우 결과는 예상과 다르게 나타나는 때가 많다.

그림 3. 2를 보자.

```
]CALL-151
*300 : FF 00 00 00 00 00
*300L
300-FF     ???
301-00     BRK
302-00     BRK
*300G
305- A=00 X=00 Y=FE ………
```

그림3. 2 6502 미정의 명령의 실행

300번지의 FF는 6502에서 쓰지 않는 명령이므로 당연히 무시되고 그 뒤의 00에 의해 브레이크가 걸려 303번지를 표시해야 하는데, 305번지가 표시되었다. 이는 301, 302번지의 두 개 명령이 무시되었다는 것을 뜻한다. 왜 그럴까? 여러 프로그래머들이 알아낸 바에 의하면 6502는 표준 명령어 외에도 실행되는 명령어가 여럿 있다는 것이다. 물론 이는 일반적인 것이 아니고 모든 CPU가 그런 것은 아니라서 이런 명령들을 사용하는 것은 불안하기 짝이 없다. 이것은 단점으로 볼 수 있다. 이것 때문에 정상적인 프로그래밍이 방해를 받기 때문이다.

참고로 다음은 6502 미정의 명령어의 리스트이다.

명칭과 설명	기 능	어드레싱 모드	어셈블러	OP-코드	N	#	NZCV
ANDX	A&X-M	Zero Page	ANDX $aa	87	3	2	----
ANDX A with X		Zero Page, Y	ANDX $aa, Y	97	4	2	
		Absolute	ANDX $aaaa	8F	4	3	
DCMP	A-(DECM)	Zero Page	DCMP $aa	C7	8	2	???-
Decrement M,		Zero Page, X	DCMP $aa, X	D7	10	2	
Compare to A		Absolute	DCMP $aaaa	CF	10	3	
		Absolute, X	DCMP $aaaa, X	DF	11*	3	
ISBC	A-(INCM)-C-	Zero Page	ISBC $aa	E7	8	2	????
Increment M,	A, C	Zero Page, X	ISBC $aa, X	F7	10	2	
Subtact from A		Absolute	ISBC $aaaa	EF	10	3	
		Absolute, X	ISBC $aaaa, X	FF	11*	3	
LDAX	M-A,M-X	Zero Page	LDAX $aa	A7	3	2	??--
Load A and X		Zero Page, Y	LDAX $aa, Y	B7	4	2	
		Absolute	LDAX $aaaa	AF	4	3	
		Absolute, Y	LDAX $aaaa, Y	BF	4*	3	
RLAN	(ROLM)&A-A	Zero Page	RLAN $aa	27	8	2	???-

명칭과 설명	기 능	어드레싱 모드	어셈블러	OP-코드	N	#	NZCV
Rotate M left,		Zero Page, X	RLAN $aa, X	37	10	2	
AND with A		Absolute	RLAN $aaaa	2F	10	3	
		Absolute, X	RLAN $aaaa, X	3F	11*	3	
RRAD	(RORM)+A+	Zero Page	RRAD $aa	67	8	2	????
Rotate M right,	C−A, C	Zero Page, X	RRAD $aa, X	6F	10	3	
Add with carry		Absolute	RRAD $aaaa	6F	10	3	
		Absolute, X	RRAD $aaaa, X	7F	11*	3	
SLOR	(ASLM)VA−A	Zero Page	SLOR $aa	07	8	2	???−
Shift M left,		Zero Page, X	SLOR $aa, X	17	10	2	
OR with A		Absolute	SLOR $aaaa	0F	10	3	
		Absolute, X	SLOR $aaaa, X	1F	11*	3	
SREO	(LSRM)VA−A	Zero Page	SREO $aa	47	8	2	???−
Shift M right,		Zero Page, X	SREO $aa, X	57	10	2	
Exclusive OR		Absolute	SREO $aaaa	4F	10	3	
with A		Absolute, X	SREO $aaaa, X	5F	11*	3	
TSTA Test bit 2in A	A& #$04−A	Absolute	TSTA $aaaa	9F	4	3	−−−−
TSTX Test bit 2in X	X& #$40−A	Absolute	TSTX $aaaa	9E	4	3	−−−−

용어설명

A 어큐물레이터	+ 더하기	$aa 8비트 제로페이지 주소	NZCV 영향을 받는 플래그
X, Y 인덱스레지스터	− 빼 기	$aaaa 16비트 절대 주소	− 영향 없음
M 메모리	& AND	N 실행 시간(cycle=10^{-6}초)	? 영향 받음
C 캐리 플래그	V EOR	# 소요 메모리	
	A−B A의 내용을 B에 넣음	* 페이지를 초과하여 일어나면 1사이클 추가	
	V OR		

표 3. 2 6052 미정의 명령어

③ 구식의 제조 공법으로 제조되고 있으므로 신뢰도가 떨어진다.

(2) 소프트웨어적 문제

① 간접 어드레싱의 JMP(6C)명령에 문제가 있다.

다음 그림을 보자.

```
]CALL-151
*300 : 6C FF 1F 0=JMP ($1FFF)
*1FFF : BF 9D
*1F00 : 00
*00BF : 00
*300G
00C1- A=02 X=3F Y=9E ………
```

그림 3. 3 잘못된 JMP의 실행

300번지의 명령은 JMP($1FFF)이다. 이는 앞에서 설명한 것과 같이 $1FFF, $2000번지에서 실효 어드레스를 취하게 되고, 따라서 실효 어드레스는 $9DBF가 되어 이 곳으로 점프하면 베이직 모드로 돌아오도록 되어 있다. 그런데 실제로는 엉뚱하게 C1번지에서 브레이크가 걸렸다. 이것은 $BF번지가 00이기 때문이다. 즉 JMP $00BF가 실행되었다는 말이다. 왜 이럴까? 이유는 JMP($1FFF)가 $1FFF와 $1F00에서 어드레스를 읽어왔기 때문이다. 이렇게 되면 $BF번지로 점프하게 된다. 이것은 명백한 CPU의 버그이다.

② 10진 플래그에 문제가 있다. 6502가 10진 플래그를 세트하여 BCD연산을 한다는 것은 앞에서 이미 설명하였다. 그런데 여기서 다시 2진 모드로 돌아갈 때에는 10진 플래그를 리셋해야 한다. 그런데 리셋을 해도 되지 않는 경우가 있다. 또한 Z, N, V 플래그도 오동작을 한다.

③ 인덱스 어드레싱에서 페이지 경계선($??FF)을 어드레싱할 때에 간혹 잘못된 어드레스를 읽어온다.

④ BRK명령이 실행된 뒤에 인터럽트가 있으면 이 명령이 무시되면서 인터럽트 벡터가 로드된다.

이런 문제들이 여태까지 애플 프로그래머를 괴롭혀 왔다. 그렇기 때문에 이런 명령을 아예 사용하지 않든가 아니면 몇 단계 더 거쳐서 처리해야 하는 경우도 많았다. 그러면 과연 이 문제가 65C02에서는 어떻게 해결되었는가?

2. 65C02에서 개량된 점

① 명령어와 어드레싱 모드의 추가 : 두 개의 어드레싱 모드, 10개의 니모닉, 27개의 OP코드가 추가되었다.

② CMOS공법으로 제조되어 재래의 NMOS로 제조된 것보다 신뢰도가 높고 전력 소비가 적다.

③ 컴퓨터 외의 잡음, 예를 들어 방해 전파 등에 대한 저항성이 높다.

④ 명령어의 버그를 개선했다.

- JMP($????)의 경우—정상적으로 움직인다.
- 10진 플래그의 경우—확실한 동작

· BRK와 인터럽트−100% O.K.

다음 표는 6502와 65C02와의 차이점과 개선점을 나타낸 것이다.

	기 능	6502	65C02
기능적인 향상과 버그의 제거	페이지 경계색을 가로지르는 인덱스 어드레싱.	부정확한 어드레스를 다시 읽음.	마지막 명령 바이트를 다시 읽음.
	부정확한 오퍼레이션 코드의 실행.	때로 리셋에 의해서만 정지. 결과는 예측 불허.	모두 NOP로 처리됨.
	간접 점프,(오퍼랜드=nnFF)	페이지가 증가 않음.	페이지가 증가하고 1사이클 추가.
	유효한 어드레스에서의 읽기 / 수정 / 써넣기.	한 번 읽고 두 번 쓰기.	두 번 읽고 한 번 쓰기.
	10진 플래그.	리셋 이후에 상태 불명.	리셋과 인터럽트 이후에는 2진모드 (D=0)로 초기화됨.
	10진 연산 이후의 플래그들.	N, V, Z 플래그에 정확한 값.	1사이클 추가로 정확한 플래그값.
	BRK 명령 해독 이후의 인터럽트.	인터럽트 벡터가 로드되고 BRK는 무시됨.	BRK가 실행되고 이어서 인터럽트가 수행됨.
하드웨어적인 개선	쓰기 동작중의 대기(RDY)명령.	무시	O_2동안 정지.
	사용 않는 입력 전용의 핀들.	노이즈 문제를 예방하기 위하여 반드시 낮은 임피던스에 연결.	고저항(약 250킬로옴)에 의해서 V_{DD}로 내부적 연결.

표 3. 3 6502와 65C02의 차이

3. 65C02에서 추가된 점

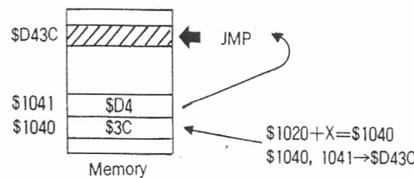
그러면 여기서는 65C02에서 추가된 점을 몇 가지 알아보자. 먼저 두 개의 어드레싱 모드가 추가되었는데, 그에 대해 상세히 알아보자.

(1) 추가된 어드레싱 모드의 사용예

① 절대 인덱스 간접 어드레싱 모드(absolute indexed indirect addressing)

오직 JMP명령에서만 사용되며, 3바이트의 명령이다. 2, 3바이트의 내용과 X 레지스터의 값을 더해서 어드레스를 구한 다음 그 어드레스에 있는 내용을 실효값으로 하여 그 번지로 점프한다.

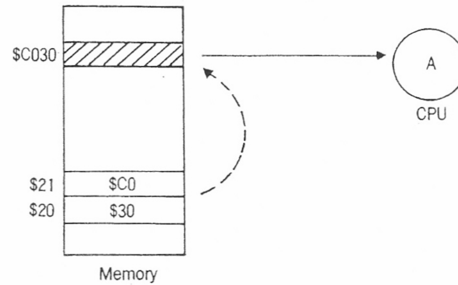
예 JMP ($1020, X)(X = 20) { $1020과 X값을 더한다. (=$1040)
$1040과 $1041의 값을 유효 어드레스로 한다. 즉 JMP $D43C와 같이 된다. }



② 제로페이지 간접 어드레싱(zero page indirect addressing)

2바이트의 명령으로, 두번째 바이트를 어드레스로 하여 그 번지값을 읽은 다음 이것을 유효 어드레스로 한다.

예 LDA ($20) { 20, 21번지의 값을 어드레스로 한다. 그것을 어드레스($C030)로 하여 LDA.



(2) 추가된 명령어의 기능과 형식

다음 표는 추가된 OP코드를 한눈에 볼 수 있게 만들어 놓은 것이다. 앞에서 소개한 명령어와 어드레싱 모드의 추가 이외에도 기존의 명령어가 이전에는 사용되지 못했던 어드레싱 모드에서 사용되는 것을 볼 수가 있을 것이다. 또한 속도가 빨라진 명령어도 있다. 이것은 처리 시간을 1사이클 줄였기 때문인데, 그것 때문에 65C02는 6502보다 동작 속도가 빠르다.

이 추가된 명령어나 속도가 증가된 명령어들은 프로그래밍을 하는 데 큰 즐거움을 줄 것이다. 그러

구분	니모닉	어드레싱 모드	OP코드
추가된 니모닉	BRA	$????	80 ??
	PHX		DA
	PLX		FA
	PHY		5A
	PLY		7A
	STZ	$??	64 ??
	STZ	$??, X	74 ??
	STZ	$????	9C ?? ??
	STZ	$????, X	9E ?? ??
	TRB	$??	14 ??
	TRB	$????	1C ?? ??
	TSB	$??	04 ??
	TSB	$????	0C ?? ??

표 3. 4 추가된 명령어

구분	니모닉	어드레싱 모드	OP코드
기존 니모닉의 확장	ADC	($??)	72 ??
	AND	($??)	32 ??
	BIT	#$??	89 ??
	BIT	$??, X	34 ??
	BIT	$????, X	3C ?? ??
	CMP	($??)	D2 ??
	DEC		3A
	EOR	($??)	52 ??
	INC		1A
	JMP	($????, X)	7C ?? ??
	LDA	($??)	B2 ??
	ORA	($??)	12 ??
	SBC	($??)	F2 ??
	STA	($??)	92 ??

명령어	어드레싱 모드	코드	사이클수 변화
ASL	$??, X	1E	7→6
DEC	$??, X	DE	7→6
INC	$??, X	FE	7→6
LSR	$??, X	5E	7→6
ROL	$??, X	3E	7→6
ROR	$??, X	7E	7→6

표 3. 5 속도의 증가

면 이번에는 각 명령어의 기능을 상세히 보기로 하자. 지면 관계상 65C02에서 추가된 것과 추가된 명령어를 포함한 명령어군에 한한다. 여기서 상태 레지스터 밑에 있는 별표(*)는 이 플래그가 명령어 사용 이후에 변동되는 것을 나타내며, 하이픈(—)으로 나타낸 것은 변동이 없는 것을 가리키고, 명령어군인 경우에는 #표가 붙은 것만이 65C02 추가 명령이다.

ADC— ADd memory to accumulator with Carry

작동 : 메모리의 것과 어큐뮬레이터와 캐리(carry)의 합을 구한 다음 그것을 어큐뮬레이터에 넣는다.

N	V			D	I	Z	C
*	*			—	—	*	*

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	ADC ($????, X)	$61	97	2
제로페이지	ADC $??	$65	101	2
직접 어드레싱	ADC #$??	$69	105	2
절대 어드레싱	ADC $????	$6D	109	3
(간접), Y	ADC ($??), Y	$71	113	2
# (제로페이지)	ADC ($??)	$72	114	2
제로 페이지, X	ADC $??, X	$75	117	2
절대, Y	ADC $????, Y	$79	121	3
절대, X	ADC $????, X	$7D	125	3

AND – logically AND memory with accumulator

작동 : 메모리의 것과 어큐뮬레이터와 논리곱(AND) 연산을 한 다음 그것을 어큐뮬레이터에 넣는다.

N	V			D	I	Z	C
*	—			—	—	*	—

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	AND ($??, X)	$21	33	2
제로페이지	AND $??	$25	37	2
직접 어드레싱	AND #$??	$29	41	2
절대 어드레싱	AND $????	$2D	45	3
(간접), Y	AND ($??), Y	$31	49	2
# (제로페이지)	AND ($??)	$32	50	2
제로페이지, X	AND $??, X	$35	53	2
절대, Y	AND $????, Y	$39	57	3
절대, X	AND $????, X	$3D	61	3

BIT – test BITs 6 and 7 in memory

작동 : 명령어 뒤의 메모리와 어큐뮬레이터와 논리곱(AND)연산을 한다. 그 결과중 일곱번째 비트를 N에, 여섯번째 비트를 V에 넣는다. 만일 그 연산 결과의 바이트가 0이면 Z가 1로 된다.

㈜ 직접 어드레싱의 BIT($89)는 위의 것과 같이 Z비트에 영향을 주지만, N과 V비트는 변경되지 않는다.

N	V			D	I	Z	C
B6	B7			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
제로페이지	BIT $??	$24	36	2
절대 어드레싱	BIT $????	$2C	44	3
# 제로페이지, X	BIT $??, X	$34	52	2
절대, X	BIT $????, X	$3C	60	3
직접 어드레싱	BIT #$??	$89	137	2

BRA – BRAnch

작동 : 조건없이 건너간다.

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 렐러티브	BRA	$80	128	2

CMP – CoMPare memory and accmulator

작동 : 명령어 뒤의 메모리와 어큐뮬레이터를 비교한 뒤 그 결과를 갖고 각 플래그를 세트한다.

N	V			D	I	Z	C
*	–			–	–	*	*

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	CMP ($??, X)	$C1	193	2
제로페이지	CMP $??	$C5	197	2
직접 어드레싱	CMP #$??	$C9	201	2
절대 어드레싱	CMP $????	$CD	205	3
(간접), Y	CMP ($??), Y	$D1	209	2
# (제로페이지)	CMP ($??)	$D2	210	2
제로페이지, X	CMP $??, X	$D5	213	2
직접, Y	CMP $????, Y	$D9	217	3
직접, X	CMP $????, X	$DD	221	3

DEC – DECrement the content memory by 1

작동 : 명령어 뒤의 메모리에서 1을 뺀다.
그 결과에 따라 N, Z 플래그가 바뀐다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	DEC	$3A	58	1
제로페이지	DEC $??	$C6	198	2
절대 어드레싱	DEC $????	$CE	206	3
제로페이지, X	DEC $??, X	$D6	214	2
절대, X	DEC $????, X	$DE	222	3

EOR – Exclusive OR memory with accumulator

작동 : 명령어 뒤의 메모리와 어큐뮬레이터와 배타적 논리합(EOR)을 하여 그 결과를 어큐뮬레이터에 넣는다. N, Z플래그를 변화시킨다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	EOR ($??, X)	$41	65	2
제로페이지	EOR $??	$45	69	2
직접 어드레싱	EOR #$??	$49	73	2
절대 어드레싱	EOR $????	$4D	77	3
(간접), Y	EOR ($??), Y	$51	81	2
# (제로페이지)	EOR ($??)	$52	82	2
제로페이지, X	EOR $??, X	$55	85	2
직접, Y	EOR $????, Y	$59	89	3
직접, X	EOR $????, X	$5D	93	3

INC – INCrement memory

작동 : 메모리의 값을 1 늘린다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	INC	$1A	26	1
제로페이지	INC $??	$E6	230	2
절대 어드레싱	INC $????	$EE	238	3
제로페이지, X	INC $??, X	$F6	246	2
절대, X	INC $????, X	$FE	254	3

JMP – JuMP unconditionally

작동 : 새로운 어드레스로 점프한다(메모리의 하위 바이트를 PCL에, 상위 바이트를 PCH에 넣는다).

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
절대 어드레싱	JMP $????	$4C	76	3
(간접 절대)	JMP ($????)	$6C	108	3
# (절대, X)	JMP ($????, X)	$7C	124	3

LDA – LoaD Accumulator with memory

작동 : 명령어 뒤의 메모리를 어큐뮬레이터에 로드한다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	LDA ($??, X)	$A1	161	2
제로페이지	LDA $??	$A5	165	2
직접 어드레싱	LDA #$??	$A9	169	2
절대 어드레싱	LDA $????	$AD	173	3
(간접), Y	LDA ($??), Y	$B1	177	2
# (제로페이지)	LDA ($??)	$B2	178	2
제로페이지, X	LDA $??, X	$B5	181	2
절대, Y	LDA $????, Y	$B9	185	3
절대, X	LDA $????, X	$BD	189	3

ORA – OR memory with Accumulator

작동 : 메모리와 어큐뮬레이터 간에 논리합(OR)연산을 한다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	ORA ($??, X)	$01	1	2
제로페이지	ORA $??	$05	5	2
직접 어드레싱	ORA #$??	$09	9	2
절대 어드레싱	ORA $????	$0D	13	3
(간접), Y	ORA ($??), Y	$11	17	2
# (제로페이지)	ORA ($??)	$12	18	2
제로페이지, X	ORA $??, X	$15	21	2
절대, Y	ORA $????, Y	$19	25	3
절대, X	ORA $????, X	$1D	29	3

PHX – PusH X register onto stack

작동 : X레지스터의 내용을 스택에 밀어넣는다.

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	PHX	$DA	218	1

PHY – PusH Y register onto stack

작동 : Y레지스터의 내용을 스택에 밀어넣는다.

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	PHY	$5A	90	1

PLX – PuL1 X register from the stack

작동 : 스택에서 데이타를 꺼내어 X레지스터에 넣는다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	PLX	$FA	250	1

PLY – PuL1 Y register from the stack

작동 : 스택에서 데이타를 꺼내어 Y 레지스터에 넣는다.

N	V			D	I	Z	C
*	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 임플라이드	PLY	$7A	122	1

SBC – SuBtract with Carry as borrow bit

작동 : 어큐뮬레이터에서 명령어 뒤의 메모리만큼 빼서 그 결과를 다시 어큐뮬레이터에 넣으며, 버로우가 있으면 캐리를 리셋한다.

N	V			D	I	Z	C
*	*			–	–	*	*

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	SBC ($??, X)	$E1	225	2
제로페이지	SBC $??	$E5	229	2
직접 어드레싱	SBC #$??	$E9	233	2
절대 어드레싱	SBC $????	$ED	237	3
(간접), Y	SBC ($??), Y	$F1	241	2
# (제로페이지)	SBC ($??)	$F2	242	2
제로페이지, X	SBC $??, X	$F5	245	2
절대, Y	SBC $????, Y	$F9	249	3
절대, X	SBC $????, X	$FD	253	3

STA – STore Accumulator to memory

작동 : 어큐뮬레이터의 내용을 정해진 메모리에 넣는다.

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
(간접, X)	STA ($??, X)	$81	129	2
제로페이지	STA $??	$85	133	2
절대 어드레싱	STA $????	$8D	141	3
(간접), Y	STA ($??), Y	$91	145	2
# (제로페이지)	STA ($??)	$92	146	2
제로페이지, X	STA $??, X	$95	149	2
절대, Y	STA $????, Y	$99	153	3
절대, X	STA $????, X	$9D	157	3

STZ – STore Zero at memory

작동 : 정해진 메모리에 제로($00)를 넣는다.

N	V			D	I	Z	C
–	–			–	–	–	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 제로페이지	STZ $??	$64	100	2
# 제로페이지, X	STZ $??, X	$74	116	2
# 절대 어드레싱	STZ $????	$9C	156	3
# 절대, X	STZ $????, X	$9E	158	3

TRB – Test memory with accumulator and Reset Bits

작동 : 메모리의 것과 어큐뮬레이터의 1비트 패턴을 비교한다. 그 결과로 Z비트가 변하고, 어큐뮬레이터의 1비트가 리셋된다.

N	V			D	I	Z	C
–	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	10진	바이트 수
# 제로페이지	TRB $??	$14	20	2
# 절대 에드레싱	TRB $????	$1C	28	3

TSB – Test memory with accumulator and Set Bits

작동 : 메모리의 것과 어큐뮬레이터의 1비트 패턴을 비교한다. 그 결과로 Z비트가 변하고 어큐뮬레이터의 1비트가 세트된다.

N	V			D	I	Z	C
–	–			–	–	*	–

어드레싱 모드	어셈블러 형식	OP 코드 16진	OP 코드 10진	바이트 수
# 제로페이지	TSB $??	$04	4	2
# 절대 어드레싱	TSB $????	$0C	12	3

§4. 애플 CPU의 한계를 넘는다.

여기서는 6502의 속도를 증가시키는 카드(액셀레이터 카드)와 6502계열의 16비트 CPU 65C816에 대해 알아보기로 하자. 단, 이것은 국내에 보급된 애플 Ⅱe의 사정에 비추어 볼 때 범주를 넘어서는 것이므로 간단히 설명하겠다.

1. CPU 액셀레이터 카드

앞에서도 잠깐 설명하였지만 6502A 등의 개량 CPU가 빠른 클럭 속도로 동작할 수 있음에도 불구하고 그렇게 동작시켜 주지 못한 이유는 여러가지 부수적인 회로가 갖추어지지 않기 때문이다. 그래서 이들 부수 회로를 포함하여 하나의 인터페이스 카드 형식으로 제조된 것이 바로 액셀레이터 카드이다. 몇 종류는 우리나라에도 들어와 있는 것 같은데, 대부분은 현재 국내에서는 구할 수 없는 것들이다.

현재 애플용 액셀레이터 카드는 인터페이스 카드 형식의 것(슬롯에 꽂아 쓴다)과 최근에 개발된 원칩 형식(CPU를 뽑고 그 곳에 넣는다)의 두 종류가 있다. 인터페이스 카드 형식은 고속(3.6 또는 4MHz)의 65C02로 마더보드의 CPU를 동작시키고, 마더보드의 램 대신에 카드 만의 고속 램으로 동작시키는 형식이다. 이들은 대부분 8핀 DIP(Dual In-line Package) 스위치를 한 개 이상 가지고 있어서 N번 슬롯 카드의 ZK EPROM(메모리에서 $Cn00~$CnFF의 어드레스를 가진다) 이 제공하는 웨어 로더(ware loader) 영역을 고속화시키는 것이 일반적이다. 스위치의 사용 방법은 먼저 DIP 스위치의 N+1번 스위치를 켜는 것이다.

원칙적으로 액셀레이터는 애플의 전기종에 대한 호환이 있으나, 카드 내부의 고속 램이 80K로 메인 64K만을 고속화시켜 주는 타이탄 테크놀로지(titan technology)의 액셀레이터도 있다. 이것은 3.6MHz의 65C02와 80KB 고속 램, I/O레벨에서의 시스템 속도에 맞추어주기 위한 저속화 회로까지 갖추고 있다.

또한 이밖에 고속 램을 가진 Mc.T사의 "Speedamon"(나중에 이것은 64K의 고속 램에서 최대 2메

가까지 확장 가능한 "Fast Ram"이란 추가 보드로 강화하였음)이 있다. 그리고 어플라이드 엔지니어링의 "TransWarp"카드와 같이 메인 64K를 위한 128K와 보조 64K를 위한 128K(모두 256K), 8 DIP 스위치를 두 개 가지는 128K 애플 Ⅱe를 위한 보드가 있다. Speedamon을 제외하고 모두 로크웰의 65C02를 사용한다고 한다.

후자의 칩 형식의 제품으로는 ZIP 테크놀로지사의 Zip Chip이 있다. 이것은 4MHz의 65C02에서 그 게이트 어레이를 변경하여 고속화할 수 있는 메모리를 최대 2메가바이트까지 바꾸게 해준다고 한다. 생산된 지 얼마 되지 않기는 하나 그 기능의 우수성으로 화제를 모으는 것이다.

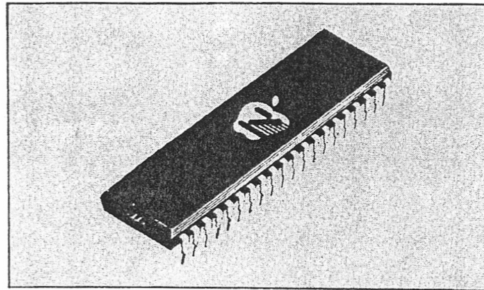


그림 3. 4 Zip Chip

2. 16비트 애플Ⅱe(65C816)

애플의 6502가 직접 억세스할 수 있는 영역은 고작 64K뿐이다. 현재 Ⅱe가 128K이긴 하지만, 이것은 뱅크 스위칭 방식을 써서 양쪽 64K로 나눠서 쓰고 있는 것에 불과한 것이다.

그러나 이러한 사용 방법에는 한계가 있고, 이 한계를 극복하기 위해서 6502 계열의 새로운 CPU가 나왔다. 그 중 대표적인 것이 Ⅱ$_{GS}$에 사용되는 W65SC816(이하 65C816이라 지칭)이다. 이것은 16비트 프로세서로, 8비트 65C02 에뮬레이션(모방) 모드로 사용할 수도 있다. 2.8MHz의 클럭 속도로 동작하며 24비트 어드레스 버스를 가지는데, 이것은 최대 16메가바이트까지 메모리를 확장할 수 있게 되어있다. 13개의 기존 65C02 어드레싱 모드 이외에 11개의 새로운 어드레싱 모드를 합해서 모두 24개의 어드레싱 모드를 가지며, 91개의 인스트럭션, 255개의 명령어를 가진다. 표 3. 6은 65C816의 명령 테이블을 나타낸 것인데, 기존 65C02에서 보지 못한 명령이 많이 있을 것이다.

또한 어큐뮬레이터와 X, Y 레지스터, 스택 포인터 등이 8비트에서 16비트로 증가되었고 데이타뱅크 레지스터라는 것이 새로 생겼는데, 이것 때문에 16메가바이트까지 어드레싱을 하는 것이 가능해진 것이다. 뿐만 아니라 다이렉트 레지스터라는 것이 하나 생겼다. 그림 3. 5는 65C816의 표준 모드와 65C02 에뮬레이션 모드의 레지스터를 나타낸 것이다.

MSD	LSD 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK s 2 8	ORA(d.x) 2 6	COP s 2*8	ORA sr 2*4	TSB d 2•5	ORA d 2 3	ASL d 2 5	ORA(dl) 2*6	PHP s 1 3	ORA imm 2 2	ASL acc 1 2	PHD s 1*4	TSB a 3•6	ORA a 3 4	ASL a 3 6	ORA al 4*5	0
1	BPL r 2 2	ORA(d).y 2 5	ORA (d) 2•5	ORA(sr).y 2*7	TRB d 2•5	OOA d.x 2 4	ASL dx 2 6	ORA(dl).y 2*6	CLC imp 1 2	ORA a.y 3 4	INC acc 1*2	TCS imp 1*2	TRB a 3•6	ORA a. x 3 4	ASL a. x 3 7	ORA al. x 4*5	1
2	JSR a 3 6	AND(d.x) 2 6	JSL al 4*8	AND sr 2*4	BIT d 2 3	AND d 2 3	ROL d 2 5	AND(dl) 2*6	PLP s 1 4	AND imm 2 2	ROL acc 1 2	PLD s 1*5	BIT a 3 4	AND a 3 4	ROL a 3 6	AND al 4*5	2
3	BMI r 2 2	AND(d).y 2 5	AND (d) 2•5	AND(sr).y 2*7	BIT d.x 2•4	AND d.x 2 4	ROL d.x 2 6	AND(dl).y 2*6	SEC imp 1 2	AND a.y 3 4	DEC acc 1•2	TSC imp 1*2	BIT a. x 3•4	AND a. x 3 4	ROL a. x 3 7	AND al. x 4*5	3
4	PTI s 1 7	FOR(d.x) 2 6	WDM RESERVED	EOR sr 2*4	MVP xyc 3*7	EOR d 2 3	LSR d 2 5	EOR(dl) 2*6	PHA s 1 3	EOR imm 2	LSR acc 1 2	PHK s 1*3	JMP a 3 3	EOR a 3 4	LSR a 3 6	EOR al 4*5	4
5	BVC r 2 2	EOR(d).y 2 5	EOR (d) 2*5	EOR(sr).y 2*7	MVN xyc 3*7	EOR d.x 2 4	LSR d.x 2 6	EOR(dl).y 2*6	CLI imp 1 2	EOR a. y 3 4	PHY s 1•3	TCD imp 1*2	JMP al 3*4	EOR a. x 3 4	LSR a. x 3 7	EOR al. x 4*5	5
6	RTS s 1 6	ADC(d.x) 2 6	PER s 3*6	ADC s 2*4	STZ d 2•3	ADC d 2 3	ROR d 2 5	ADC(dl) 2*6	PLA s 1 4	ADC imm 2 2	ROR acc 1 2	RTS s 1*6	JMP(a) 3 5	ADC a 3 4	ROR a 3 6	ADC al 4*5	6
7	BVS r 2 2	ADC(d).y 2 5	ADC (d) 2•5	ADC(sr).y 2*7	STZ dx 2•4	ADC d.x 2 4	ROR d.x 2 6	ADC(dl). 2*6	SEI imp 1 2	ADC a. y 3 4	PLY s 1•4	TDC imp 1*2	JMP(a.x) 3•6	ADC a. x 3 4	ROR a. x 3 7	ADC al. x 4*5	7
8	BRA r 2•2	STA(d.x) 2 6	BRL rl 3*3	STA sr 2*4	STY d 2 3	STA d 2 3	STX d 2 3	STA(dl) 2*6	DEY imp 1 2	BIT imm 2•2	TXL imm 1 2	PHB s 1*3	STY a 34	STA a 3 4	PTX a 3 6	STA al 4*5	8
9	BCC r 2 2	STA(d).y 2 6	STA(d) 2•5	STA(sr).y 2*7	STY d.r 2•4	STX d.x 2 4	STX d.y 2 4	STA(dl). y 2*6	TYA imp 1 2	STA a. y 3 5	TXS imp 1 3	TXY imp 1*2	STZ a 3•4	STA a. x 3 5	STZ a. x 3•6	STA al. x 4*5	9
A	LDY imm 2 2	LDA(d.x) 2 6	LDX imm 2 2	LDA sr 2*4	LDY d 2 3	LDA d 2 3	LDX x 2 3	LDA(dl) 2*6	TAY imp 1 2	LDA imm 2 2	TAX imp 1 2	PLB s 1*4	LDY a 3 4	LDA a 3 4	LDX a 3 4	LDA al 4*5	A
B	BCS r 2 2	LDA(d).y 2 5	LDA(d) 2•5	LDA sr 2*7	LDY d.x 2 4	LDA d.x 2 4	LDX d.y 2 4	LDA(dl).y 2*5	CLV imp 1 2	LDA a. y 3 4	TSX imp 1 2	TYX imp 1*2	LDY a. x 3 4	LDA a. x 3 4	LDX a. y 3 4	LDA al. x CMP al	B
C	CPY imm 2 2	CMP(d.x) 2 6	REP imm 2*3	CMP sr 2*4	CPY d 2 3	CMP d 2 3	DEC d 2 5	CMP(dl) 2*6	INY imp 1 2	CMP imm 2 2	DEX imm 1 2	WAI imp 1*3	CPY a 3 4	CMP a 3 4	DEC a 3 6	4*5 CMP al. x	C
D	BNE r 2 2	CMP(d).y 2 5	CMP(d) 2•5	CMP(sr).y 2*7	PEI s 2*6	CMP d.x 2 4	DEC d.x 2 6	CMP(dl).y 2*6	CLO imp 1 2	CMP a. y 3 4	PHX s 1•3	STP imp 1*3	JML(a) 3*6	CMP a. x 3 4	DEC a. x 3 7	4*5 SBC al	D
E	CPX imm 2 2	SBC(d.x) 2 6	SEP imm 2*3	SBC sr 2*4	LPX d 2 3	SBC d 2 3	INC d 2 5	SBC(dl) 2 6	INX imp 1 2	SBC imm 2 2	NOP imp 1 2	XBL imp 1*3	CPX a 3 4	SBC a 3 4	INC a 3 6	4*5 4*5	E
F	BEO r 2 2	SBC(d.)y 2 5	SBC(d) 2•5	SBC(sr).y 2*7	PEA s 3*5	SBC d.x 2 4	INC d.x 2 6	SBC(dl).y 2*5	SED imp 1 2	SBC a. y 3 4	PLX s 1*4	XCE imp 1*2	JSP(a.x) 3*6	SBC a. x 3 4	INC a. x 3 7	SBC al. x 4*5	F

표 3. 6 65C816의 명령 테이블

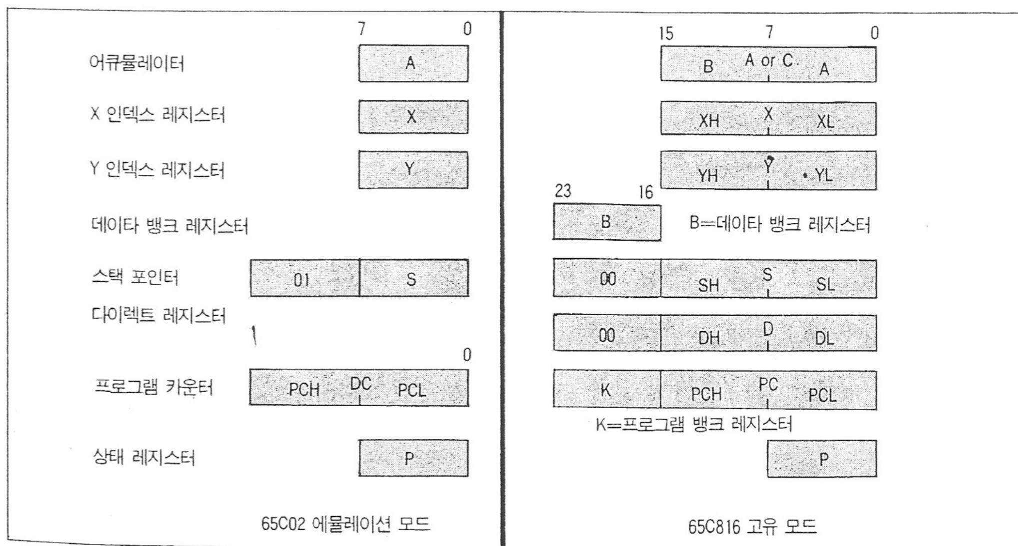


그림 3. 5 65C816 레지스터

제 4 장

128K의 관리와 사용

§1. 왜 128K가 필요한가
§2. 애플 128K 개관
§3. 메인 – 보조 메모리간의 메모리 전송 관계 ROM 루틴
§4. 메모리 관리와 관계된 소프트 스위치
§5. 128K를 능가하는 애플 IIe

현재 국내에 보급된 애플Ⅱe 기종이 128K의 RAM을 내장하고 있다는 것을 모르는 독자는 없을 것이다. 그러나 애플에 사용하고 있는 CPU는 한 번에 64K만을 어드레싱하기 때문에 이 128K를 다 사용하려면 소프트 스위치(soft switch)를 사용해야 된다. 그런데 이의 사용법이 생각보다 쉽지만은 않아서 대부분의 국내 Ⅱe 사용자가 이 128K머신을 제대로 사용하지 못하고 있는 것같다. 이에 128K를 사용하는 방법을 예제 프로그램과 함께 자세히 설명하기로 한다.

§1. 왜 128K가 필요한가

본론에 앞서 왜 128K가 우리에게 필요한지를 한번 짚고 넘어가기로 하자. 이전에 우리나라에 많이 보급된 애플 Ⅱ+는 64K(스탠다드+16K)이다. 그러나 독자 여러분들 가운데 큰 프로그램을 만들어 본, 특히 한글이 쓰여지는 프로그램을 만든 경험이 있다면 이 메모리의 부족이라는 문제에 접해 본 일이 있을 것이다. 이 자리에서 논하기에는 좀 부적당한 감이 없지 않으나 한번 생각해 보자. 기존의 한글 마스터 프로그램은 그 자체 프로그램이 $800번지부터 약 10K바이트를 사용하고, 또 $2000번지부터 $3FFF번지까지는 고해상도 페이지 1이므로 한글을 표시할 때 사용한다. 그렇기 때문에 사용자 영역은 $4000번지부터 위치되는데 $9600번지부터는 도스가 사용하므로 사용자 영역은 고작 21.5KB가

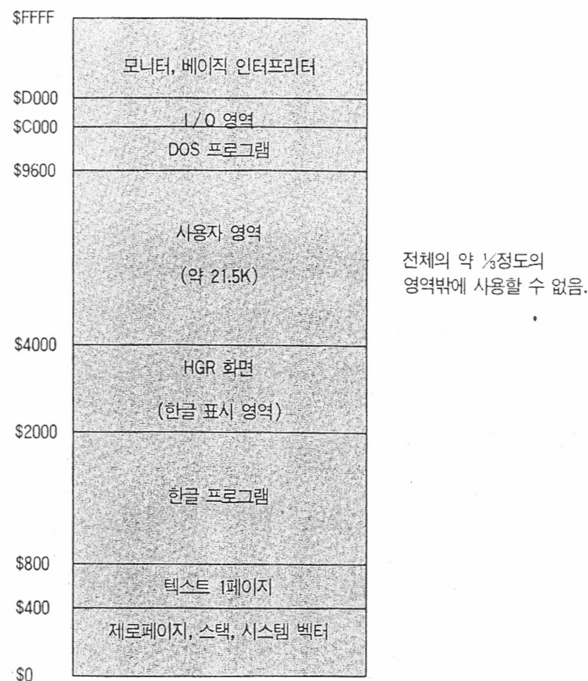


그림 4. 1 한글 사용시의 64K 메모리 맵

된다. 필자도 애플용 한글 워드프로세서 프로그램을 만들던 중에 메모리의 부족이라는 문제에 직면한 경험이 있는 터라 처음 국산 애플Ⅱe의 광고가 나갔을 때 Ⅱe의 128K는 큰 관심거리가 되었다. 참고로 그림 4.1은 64K애플에서 한글이 사용될 때의 메모리 맵인데, 여기서 사용자 영역은 전체의 약 1/3밖에 되지 않음을 알 수 있다(물론 도스를 16K 램카드로 옮기는 방법 등이 없는 것은 아니나, 이것은 어디까지나 하나의 '편법'이지 일반적인 것은 아니다. 더구나 현재 애플 컴퓨터의 표준 도스가 되어 있는 ProDOS를 쓸 때에는 이 방법조차 불가능하다).

이런 이유들 때문에 128K는 최근 퍼스널 컴퓨터 프로그램의 기능이나 용도가 더욱 다양해지는 때에 필수적인 것이라고 보아도 좋을 것이다. 이는 우리나라에서 8비트 컴퓨터의 맞수인 MSX2가 128K의 메인 램을 내장하고 있는 것을 보아도 잘 알 수 있다. 이런 현실에서 이제부터 애플Ⅱe의 128K를 한번 살펴보자.

§2. 애플 128K 개관

본론에 들어가기 전에 애플의 128K는 어떻게 쓰이고 있나를 알아볼 필요가 있다. 그림 4. 2를 보자. 이것은 애플Ⅱe의 128K의 메모리 맵이다.

1. 128K 메모리 맵과 그 용도

그림의 맨 왼쪽이 시스템 ROM이고 그림 중앙이 메인 메모리(main memory), 맨 오른쪽이 보조 메모리(aux memory)이다. 오른쪽부터 설명하면 다음과 같다.

① 시스템 모니터 및 애플소프트 인터프리터

애플소프트 베이직 인터프리터와 모니터가 위치하며, $D000번지에서 $FFFF까지의 메모리 번지를 가진다. 기존의 Ⅱ+의 것과는 약 600바이트 정도의 차이가 있고, 그 때문에 기존의 프로그램이 작동하지 않는 경우가 가끔 있다. 예를 들면 머린 어셈블러(1면짜리는 제대로 동작한다) 같은 프로그램이 오동작하는 경우도 있다.

② 시스템 모니터 확장 루틴

기존의 애플Ⅱ+에는 없던 것이다. 이것을 모니터로 덤프해 보면 그 프로그램이 보이지 않는데, 이것은 그 영역이 I/O영역과 겹치고 보통은 I/O영역이 덤프되기 때문이다. 이곳에는 미니 어셈블러와 모니터 확장 루틴이 위치한다. 메모리에서는 $C100－$C2FF번지의 영역을 가진다.

③ 80컬럼 펌웨어

$C300번지부터 위치하는 서브루틴으로서 모니터로 덤프해 볼 수 있다. 이 루틴은 메인 메모리와 보조 메모리간의 메모리 전송이라든가 80컬럼 상태에서의 화면 제어 및 늘어난 ESC 시켄스 등의 일을

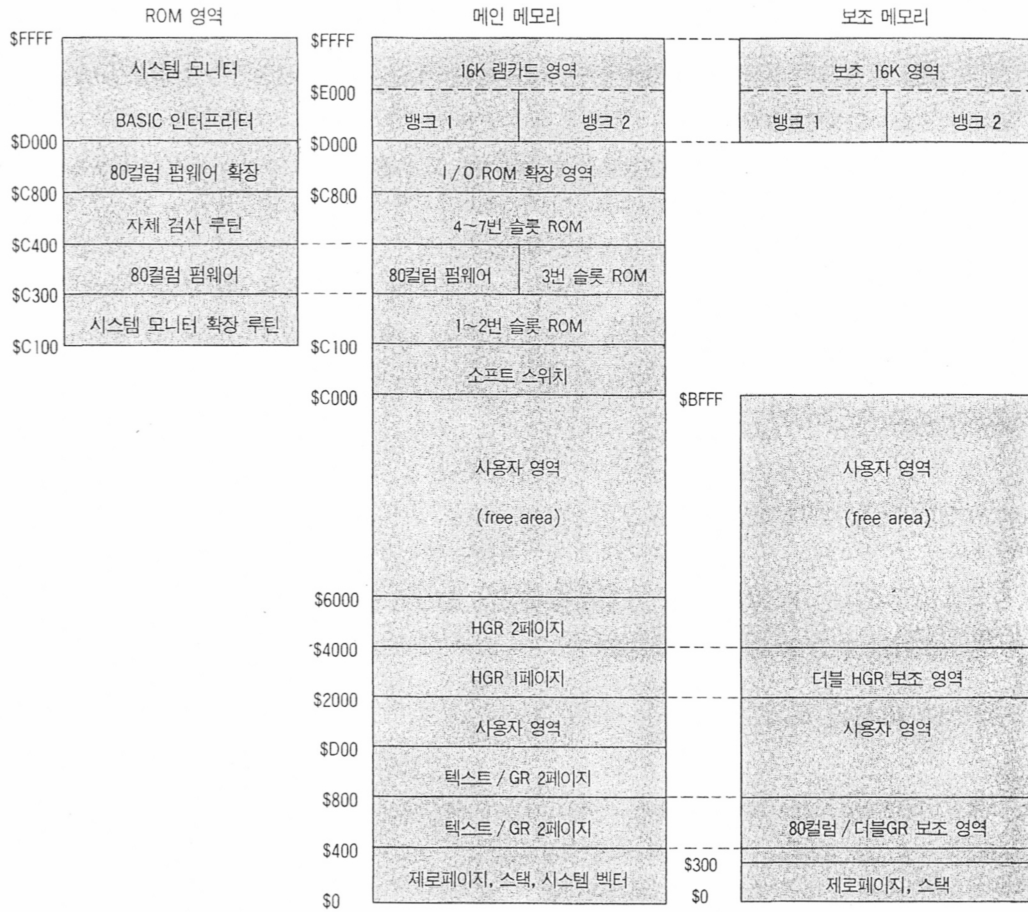


그림 4. 2 애플 Ⅱe의 128K 메모리 맵

해주는 루틴이 들어 있다. $C300 - $C3FF까지의 것과 $C800 - $CFFF까지의 확장 루틴이 있다(이 확장 루틴은 덤프해 볼 수 없다).

④ 자체 검사 루틴

어느 컴퓨터에도 없는, 애플 Ⅱe만의 독특한 기능인 자체 검사 루틴이 위치한다. 사용법은 CTRL - Close Apple - Reset을 함께 누르면 된다는 것은 이미 잘 알고 있을 것이다. 일단 이들 키가 눌려지면 컴퓨터는 그에 관련된 소프트 스위치를 ON하고 이 루틴으로 들어온다. $C400번지부터 $C7FF번지에 위치하며, 덤프해 볼 수 없다.

⑤ 메인 메모리의 랭귀지 카드 영역

기존의 Ⅱ + 의 것과 완전히 같다. $D000 - $FFFF까지이며 $D000부터 $DFFF까지는 두 개의 뱅

크로 되어있다.

⑥ I/O영역

기존의 II+와 같으며, I/O카드가 끼워져 있을 때 각각의 영역은 그 카드에 있는 ROM의 내용으로 채워진다. $C000부터 $C0FF까지는 소프트 스위치이며 $C100-$C7FF까지는 각 슬롯에 해당되는 롬인데, 3번 슬롯에 해당되는 영역은 80컬럼 펌웨어가 위치한다. 그리고 $C800번지부터 $CFFF번지까지는 ROM 확장 영역이다.

⑦ 고해상도 그래픽 영역

메인 메모리의 $2000번지부터 $5FFF까지이며, 두 개로 나뉘어져 있다. 즉 $2000번지부터 $3FFF까지의 고해상도 1페이지(HGR)와 $4000번지부터 $5FFF까지의 고해상도 2페이지(HGR2)의 두 개이다. 더블 고해상도 모드(double HGR)가 사용될 때에는 메인 메모리의 $2000-$3FFF와 보조 메모리의 $2000-$3FFF의 16K바이트가 쓰이는데, 더블 고해상도 모드에서는 한 개의 화면만이 있을 뿐이다. 더블 고해상도에 대해서는 뒤에 자세히 설명하겠다.

⑧ 텍스트/저해상도 그래픽 영역

메인 메모리의 $400번지부터 $BFF번지까지는 텍스트와 저해상도 그래픽 영역이다. 이것 또한 두 개로 나뉘어져 있다. $400-$7FF의 1페이지와 $800-$BFF의 두 페이지로 나뉘는데 보통 2페이지는 베이직 시작 번지와 겹치는 관계로 잘 쓰이지 않는다. 그리고 이 메모리가 텍스트 화면이냐 고해상도 영역이냐는 소프트 스위치를 써서 전환한다. 더블 저해상도 모드나 80컬럼 텍스트를 쓸 때에는 보조 메모리의 $400-$7FF까지의 영역과 함께 쓴다. 이것 또한 한 페이지밖에 없다.

⑨ 제로페이지/스택 영역

제로페이지는 모니터 프로그램 및 베이직 인터프리터 등에서 유용하게 쓰고 있으며, 스택은 CPU에서 JSR, PHA, PLA 등의 명령에서 사용하고 있다. 메인 메모리와 보조 메모리에 각각 있는데 보통은 메인 메모리의 것을 쓴다. 여기서 주의할 점은 메인 메모리의 이 영역과 보조 메모리의 이 영역은 그 내용이 다른데 모르고 소프트 스위치를 보조 메모리의 것으로 전환했을 때에는 이 값이 틀려지므로 프로그램이 정지하게 된다. 따라서 보조 메모리는 보통 데이타 보관 영역으로 쓰지 프로그램 영역으로는 잘 쓰지 않는다. 메모리상의 위치는 $0-$FF까지가 제로페이지, $100-$1FF까지가 스택이다.

⑩ 그밖의 영역은 사용자가 마음대로 사용할 수 있다. 단 $9600번지부터 $BFFF까지의 것은 도스가 사용한다. 또한 보조 메모리를 쓸 때에는 각별히 주의해야 한다.

2. 128K 이용 사례

이전까지는 우리나라에 128K의 애플이 선보이지 않은 관계로 128K를 사용하는 프로그램이나 128K

에서만 작동하는 프로그램이 하나도 선보이지 않았다. 또한 기존의 Ⅱ+에서 동작하는 프로그램이 Ⅱe에서는 새로운 기능을 보인다는 것을 모르고 있었을 것이다. 이 자리에서 간단히 소개하면 다음과 같다.

① ProDOS

128K에서는 보조 메모리의 64K를 램디스크로 쓴다. 이 때 볼륨 네임은 /RAM이며, 슬롯 3에 2번 드라이브처럼 인식된다. 여기서 한 가지 주의할 점은 이 램디스크 서브루틴에는 약간의 버그가 있어서 프로도스의 낮은 버전에서는 제대로 동작하지 않을 수도 있다.

② Copy Ⅱ Plus

5.1 이상부터는 보조 메모리를 데이타 영역으로 쓰기 때문에 한 번에 18개 섹터씩 읽고 쓰며, 그렇기 때문에 디스켓을 한 번 복사하는데 단 두 번만에 OK이다. 또한 프로도스 버전의 것(6.3 이상)에서는 램디스크를 쓸 수가 있어서 한 개의 드라이브로 짧은 크기의 많은 프로그램을 복사할 때에는 먼저 램디스크에 복사해 놓고 다음에 그것을 복사할 디스크로 옮기면 훨씬 효율적인 복사를 할 수가 있다.

③ Locksmith 5.0 Fast Backup

이것 또한 보조 메모리를 데이타 영역으로 사용하기 때문에 단 두 번에 복사가 가능하다.

㊟ 시중에는 두 가지의 Locksmith Fast가 나돌고 있는데, 하나는 68섹터의 원본이며 다른 하나는 40트랙을 복사할 수 있는 19섹터짜리 개조판이다. 개조판의 경우에는 애플Ⅱe에서 사용 불가능하다.

④ 엑스트라 K(Extra K)

우리에게 잘 알려진 비글 브로스의 작품이다. 보조 메모리를 다음과 같은 것의 데이타 영역으로 사용할 수 있게 만들어 준다.

– Extra Variables : 보조 메모리에 애플소프트 프로그램의 변수명을 저장해서 그만큼 프로그램 영역을 늘리면서 큰 배열변수의 사용을 가능하게 해준다.

– Extra Copy : 보조 메모리를 복사할 때의 데이타 영역으로 만들어 준다.

– Extra Screen : 보조 메모리에 텍스트, 저해상도, 고해상도 그래픽, 더블 고해상도 그래픽 화면 등을 저장해 놓고 이것들을 번갈아 보여줌으로써 슬라이드를 보는 것과 같은 효과를 낸다.

– Extra Apple : 128K의 애플을 두 개의 64K 애플로 나누어 사용자로 하여금 마치 두 대의 애플을 사용하고 있는 것처럼 만들어 준다. 한 쪽에는 도스 3.3을, 다른 쪽에는 프로도스를 돌리는 것도 가능하며, 이럴 때에는 도스 3.3과 프로도스의 파일 변환이 별도의 프로그램을 거치지 않고도 가능하다.

⑤ Disk Quick

역시 비글 브로스사의 작품으로 도스 3.3에서 보조 메모리를 램디스크로 쓸 수 있게 해준다.

⑥ Ultima V

머킹보드를 사용할 때에는 그 서브루틴과 20곡에 달하는 음악 데이타를 보조 메모리에 넣고 사용한다. 이것이 없으면 이 프로그램에서 배경 음악을 들을 수 없다.

⑦ Game Maker

프로그램의 일부 기능을 보조 메모리에 저장하여 디스켓을 교환하는 번거로움을 없앴다.

⑧ Z-Basic

128K에서는 더 많은 사용자 영역을 가진다.

⑨ 한글 워드프로세서 "나랏말씀"

보조 메모리에 한자 폰트를 넣으므로 애플에서의 한자 사용이 가능하다. 이밖에도 많은 프로그램이 있겠지만 현재의 프로그램 추세는 기능이 많고 그 길이가 길어지는 것이 보편적이므로 앞으로 더 많은 128K 사용 프로그램이 나오리라고 믿는다.

§3. 메인 - 보조 메모리간의 메모리 전송 관계 ROM 루틴

앞에서 Ⅱe의 보조 메모리가 프로그램 영역으로 사용되는 것이 아니라 데이타 영역으로 사용된다고 언급한 일이 있다. 한 번에 실행되는 프로그램의 길이는 그리 긴 것은 아니므로 한 번에 64K를 넘어가는 프로그램은 없고 그때그때 불러내서 쓰는 것뿐이다. 이럴 때 보조 메모리와 본 메모리간의 메모리 전송 기법에 대한 지식이 필요하다. 여기에서는 이것에 대해서 자세히 알아보도록 한다. 메인 메모리와 보조 메모리간에 데이타를 전송하는 방법은 크게 두 가지로 나누어 볼 수 있다. 그것은 먼저 모니터(또는 확장 롬 루틴)에 있는 서브루틴을 불러내어 쓰는 것이고, 두번째는 소프트 스위치를 적절히 사용하여서 직접 프로그램을 옮기는 프로그램을 짜는 것이다.

그래서 여기서는 그 모니터 루틴에 대해 사용법 등을 자세히 알아보려고 한다. 모니터 루틴은 모두 두 가지인데, 이것만 가지고도 그리 큰 문제없이 사용할 수 있다.

1. MOVEAUX 서브루틴($C311)

만일 여러분이 128K를 사용하는 프로그램을 작성할 때에는 이 서브루틴을 가장 많이 사용할 것이다. 그러므로 예제 프로그램과 함께 자세히 설명하겠다. 먼저 다음 프로그램을 보자.

(1) AUXMOVE 프로그램

```
]LIST
1 REM —메인 / 보조 메모리 이동 프로그램 —
2 REM    (Main / Aux. mover.)
3 REM —1989년 1월 23일
```

```
4  REM    by U.S. Yun.—
5  REM
10  FOR P=768 TO 771
20  READ D : POKE P, D : REM 서브루틴으로 점프하라는 데이타를 POKE한다.
30  NEXT P : DATA 56, 76, 17,195
40  REM   (JMP $C311)
50  TEXT : HOME
60  PRINT "메모리 전송 프로그램  /  by W.K.P."
70  PRINT : PRINT "어떤 동작을 원합니까?"
80  PRINT "1. 메인->보조 메모리 이동."
90  PRINT "2. 보조->메인 메모리 이동. ":
100  GET K$ : IF K$<"1" OR K$>"2" THEN 100
110  K=VAL(K$) : PRINT K
120  POKE 768, 88-K*32
130  REM -$C311번지로 점프할 때에는 메인에서 보조냐 보조에서 메인이냐를 선택해야 되는데,
140  REM 768번지에 56(SEC)을 넣으면 메인 -> 보조로 이동되고,
150  REM              24(CLC)를 넣으면 보조 ->메인으로 이동된다.
160  PRINT : INPUT "원래 메모리의 시작 번지는? : ";S
170  SH=INT(S / 256) : SL=S-SH*256 : REM 상위 / 하위 바이트 계산.
180  POKE 60, SL : POKE 61, SH : REM $3C, $3D번지에 시작 번지를 넣는다.
190  INPUT "원래 메모리의 끝번지는? : ";E
200  EH=INT(E / 256) : EL=E-EH*256
210  POKE 62, EL : POKE 63, EH : REM $3E, $3F번지에 끝번지를 넣는다.
220  INPUT "옮겨질 곳의 시작 번지는? : ";T
230  TH=INT(T / 256) : TL=T-TH*256
240  POKE 66, TL : POKE 67, TH : REM $42, $43번지에 이동해 갈 번지를 넣는다.
250  PRINT : PRINT "메모리 이동 시작합니다."
260  CALL 768 : REM "JMP $C311"
270  PRINT "다 되었읍니다." : END
```

리스트 4. 1 메모리 이동 프로그램

이 프로그램을 입력한 후(단, 아래의 기계어는 입력하지 않는다) 실행시키면 먼저 메인 메모리와 보

조 메모리 중 어느 곳에서 어느 곳으로 이동시킬까를 묻는다. 1을 입력하면 메인에서 보조로, 2를 입력하면 보조에서 메인으로 이동하게 된다. 그 다음에는 원래 메모리의 시작 번지와 끝번지를 입력하고 옮겨갈 메모리의 시작 번지를 입력한다. 이것으로 모든 과정은 OK! 화면에는 "다 되었읍니다"라는 메시지가 나온 후 프로그램의 수행을 멈춘다.

프로그램의 원리를 설명하면 이 MOVEAUX 서브루틴을 완전히 이해할 수 있게 된다. 먼저 약간의 기계어 프로그램을 READ-DATA문으로 메모리의 768($300)번지부터 넣는다. 아래의 기계어 덤프 리스트가 바로 그것이다. 그 다음에는 메인 메모리에서 보조 메모리로 가느냐, 보조 메모리에서 가느냐를 결정한 다음에 어떠한 값을 768번지에 POKE한다. 이는 메인에서 보조로 옮길 때에는 캐리를 세트하고 보조에서 메인으로 옮길 때에는 캐리를 리셋하는데, 768번지에 CLC, SEC명령을 넣으면 간단히 해결된다. CLC명령의 기계어 코드가 24($18)이고 SEC가 56($38)이므로 이를 넣는다.

그 다음에는 이동시킬 본 메모리의 시작 번지/끝번지를 입력받아 POKE 한다. 시작 번지는 $3C, $3D번지에 하위 바이트 먼저 들어가고 끝번지는 $3E, $3F번지에 들어간다. 한편 옮겨갈 곳의 메모리는 $42, $43번지에 들어가는데, 이것은 다음에 입력해서 넣는다. 그러면 모든 파라메터가 세트된 것이고, CALL 768로 옮긴다. 768($300)번지에는 캐리를 세트/리셋하고 메모리를 옮기는 메인 루틴인 $C311번지로 점프한다.

```
*300L
0300-38              SEC
0301-4C 11 C3        JMP $C311
0304-00              BRK
```

(2) MOVEAUX루틴의 사용예를 통한 이해

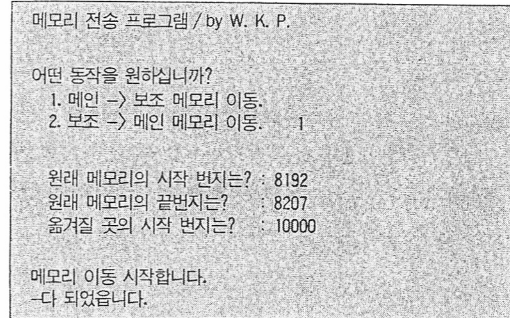
이제 한번 사용해 보자. 먼저 다음을 입력해 본다.

```
FOR A=8192 TO 8207:POKE A, A-8192:NEXT A
```

이렇게 입력하면 메모리의 8192($2000)번지부터 8207($2007)번지까지는 다음과 같다.

```
]CALL-151
*2000.200F
2000-00 01 02 03 04 05 06 07
2008-08 09 0A 0B 0C 0D 0E 0F
*
```

이제 앞의 MOVEAUX 프로그램을 실행한다. 1번을 눌러 메인에서 보조로 옮겨지게 하고 시작번지는 8192로, 끝번지는 8207로, 옮겨질 번지는 10000으로 한다. 화면에 "다 되었읍니다"라고 나오면 정상이다.


메모리 전송 프로그램 / by W. K. P.

어떤 동작을 원하십니까?
1. 메인 -> 보조 메모리 이동.
2. 보조 -> 메인 메모리 이동. 1

원래 메모리의 시작 번지는? : 8192
원래 메모리의 끝번지는? : 8207
옮겨질 곳의 시작 번지는? : 10000

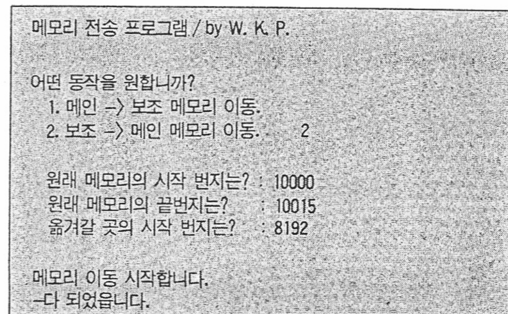
메모리 이동 시작합니다.
-다 되었읍니다.

그림 4. 3 메인→보조 메모리로 이동

그러면 메인 메모리에 8192번지부터 있던 것이 보조 메모리의 10000번지에서 10015번지까지로 이동한 것이다. 이것을 확인해 보기 위해서 다음을 입력한다.

```
]HGR : CALL - 151
*2000.200F
2000 - 00 00 00 00 00 00 00 00
2008 - 00 00 00 00 00 00 00 00
*
```

당연히 HGR이라는 명령을 실행했으므로 메모리는 지워지게 마련이다. 하지만 앞에서 이 부분을 보조 메모리에 옮긴 적이 있다. 이것을 원래 자리로 가져오자. 앞의 프로그램을 실행시키고 다음을 입력한다. 먼저 2번을 눌러 보조 메모리에서 메인 메모리로 옮기게 하고 시작 번지는 10000, 끝번지는 10015, 옮겨질 곳은 8192로 한다.


메모리 전송 프로그램 / by W. K. P.

어떤 동작을 원합니까?
1. 메인 -> 보조 메모리 이동.
2. 보조 -> 메인 메모리 이동. 2

원래 메모리의 시작 번지는? : 10000
원래 메모리의 끝번지는? : 10015
옮겨갈 곳의 시작 번지는? : 8192

메모리 이동 시작합니다.
-다 되었읍니다.

그림 4. 4 보조→메인 메모리로 이동

"다 되었읍니다"라는 말이 나오면 다음과 같이 확인한다.

```
]CALL - 151
*2000.200F
2000 - 00 01 02 03 04 05 06 07
2008 - 08 09 0A 0B 0C 0D 0E 0F
*
```

이 프로그램을 이용하면 보조 메모리와 메인 메모리와의 프로그램 및 데이타 전송을 마음대로 할 수 있다.

(3) MOVEAUX / 프로그램 요약

MOVEAUX 서브루틴

기　　능 : 메인/보조 메모리간의 메모리 전송.

사 용 법 : 파라메터를 세트하고 JSR $C311과 같이 한다.

레지스터 : A, X, Y 레지스터 모두 보존된다.

파라메터 : 캐리(C)-세트(C=1)일 때 : 메인→보조 메모리로 이동.

리셋(C=0)일 때 : 보조→메인 메모리로 이동.

$3C, $3D번지 : 이동될 시작 번지를 넣음.

$3E, $3F번지 : 이동될 끝번지를 넣음.

$42, $43번지 : 이동해 갈 곳의 시작 번지.

㈜ 이 MOVEAUX 서브루틴에서 옮겨질 수 있는 메모리는 $200-$BFFF번지까지에 한하며, 제로페이지나 스택 또는 16K램카드나 ROM의 영역의 것을 이동시키려는 생각은 금물이다.

2. XFER 서브루틴($C314)

MOVEAUX 서브루틴과 함께 곧잘 쓰이는 중요한 보조 메모리 관리 서브루틴이 바로 XFER 서브루틴이다. 이것의 의미나 사용 방법이 약간 생소하겠지만 잘 써보면 쉽게 마스터할 수 있을 것이다.

XFER 서브루틴은 메인 메모리와 보조 메모리간에 프로그램의 제어를 옮기는 기능을 한다. 즉 CPU는 보통 때에는 메인 메모리의 프로그램을 제어하고, 이 서브루틴을 사용하여 CPU의 제어를 보조 메모리로 옮길 수도 있는 것이다. 프로그램을 보면서 이 서브루틴의 확실한 사용법을 알아보도록 하자.

(1) XFER 서브루틴의 사용예

```
]CALL-151
*300L
300-A9 32      LDA  #$32
302-85 3C      STA  $3C
304-A9 03      LDA  #$03
306-85 3D      STA  $3D       ; 시작 번지= $332
308-A9 48      LDA  #$48
30A-85 3E      STA  $3E
30C-A9 03      LDA  #$03
30E-85 3F      STA  $3F       ; 끝번지= $348
310-A9 00      LDA  #$00
312-85 42      STA  $42
314-A9 08      LDA  #$08
316-85 43      STA  $43       ; 옮겨갈 번지= $800
318-38         SEC            ; 메인 -> 보조로
319-20 11 C3   JSR $C311      ; MOVEAUX 서브루틴
31C-A9 00      LDA  #$00
31E-8D ED 03   STA  $3ED
321-A9 08      LDA  #$08      ; 제어가 옮겨질 번지= $800
323-8D EE 03   STA  $3EE      ; (보조 메모리의 것이다.)
326-BA         TSX
327-86 FF      STA $FF        ; 현재의 스택 포인터를 보존한다.
329-38         SEC            ; 메인에서 보조의 것으로 제어 이동.
32A-B8         CLV            ; 제로페이지 등은 메인을 사용함.
32B-4C 14 C3   JMP $C314      ; XFER 서브루틴.
32E-A6 FF      LDX $FF
330-9A         TXS            ; 스택 포인터를 원상 복귀.
331-60         RTS            ; 프로그램의 수행을 마침.
332-A2 08      LDX  #$08      ; 수행 횟수= 8회
334-20 DD FB   JSR $FBDD      ; 벨을 울린다.
```

```
337-CA          DEX
338-D0 FA       BNE $334      : 모두 8회를 울린다.
33A-A9 2E       LDA #$2E
33C-8D ED 03    STA $3ED
33F-A9 03       LDA #$03      : 제어가 옮겨질 번지= $32E
341-83 EE 03    STA $3EE      : (단 보조 메모리의 것이다.)
344-18          CLC           : 보조에서 메인 메모리로 제어를 옮김.
345-B8          CLV           : 제로페이지 등은 계속 메인의 것으로.
346-4C 14 C3    JMP $C314     : XFER 서브루틴으로.
```

리스트 4·2 프로그램 제어 이동 예제 프로그램

(2) 프로그램의 설명과 동작 원리

본 프로그램은 크게 네 부분으로 나누어져 있다. 이것을 하나씩 설명하면 다음과 같다.

$300-$31B번지

MOVEAUX 서브루틴을 사용하여 메인 메모리의 $332부터$348번지까지를 보조 메모리의 $800번지 이하로 옮긴다.

$31C-$339번지

이 서브루틴은 메인 메모리에서 보조 메모리로 제어를 옮기는 역할을 하게 되는 XFER루틴을 사용하는 곳이다. $3ED, $3EE번지에는 제어를 옮길 곳의 메모리 번지를 가진다. 여기서 $800번지는 하위 바이트가 먼저($3ED : 00 $3EE : 08) 들어있다. 그 다음에는 현재의 스택 포인터를 보존한다(TSX/STX $FF). 캐리 플래그를 세트하는 것(SEC)은 메인 메모리에서 보조 메모리로 제어를 옮긴다는 뜻이고 오버플로우 비트를 클리어하는 것(CLV)은 보조 메모리로 제어를 옮겨도 제로페이지와 스택 및 16K램($D000-$FFFF) 부분은 메인 메모리의 제어를 그대로 쓰겠다는 것이다. 이것이 끝나면 보조 메모리의 $800번지로 제어를 옮긴다. 다시 제어가 메인 메모리로 돌아오면 스택 포인터를 복구하고 프로그램의 수행을 끝낸다.

$332-$339번지

이 루틴을 위해서 임의로 만든 서브루틴이다. 이것은 모니터 롬의 BELL 루틴($FBDD)을 여덟 번 CALL한다. 즉 여덟 번 연속으로 벨이 울리게 된다. 앞에서 모니터와 상부 16K를 그대로 보존시킨 이유는 바로 이 모니터 루틴을 사용하기 위함이다.그러나 이 서브루틴은 보조 메모리의 $800번지 이후로 옮겨지게 되고 실지로 실행되는 것은 그 곳의 서브루틴이다.

$33A-$348번지

이 서브루틴은 보조 메모리에서 메인 메모리로 제어를 옮기는 구실을 하게 되는 XFER루틴을 사용

하는 곳이다. $3ED, $3EE번지에는 제어를 옮길 곳의 메모리 번지를 가진다. 여기서는 메인 메모리의 $32E번지로 돌아와야 하므로 $2E, $03이 들어있다. 캐리 플래그를 리셋하는 것(CLC)은 보조 메모리에서 메인 메모리로 제어를 옮기며 오버플로우 비트를 클리어하는 것(CLV)은 보조 메모리로 제어를 옮겨도 제로페이지와 스택 및 16K램($D000-$FFFF) 부분은 메인 메모리의 제어를 그대로 쓰겠다는 것이다. 이것이 끝나면 메인 메모리의 $32E번지로 제어를 옮긴다.

그러면 이 프로그램을 입력한 뒤 실행해보자. 벨이 여덟 번 울리고 프로그램의 수행을 마치면 성공이다. 수행 순서는 먼저 $300번지 이후의 서브루틴이 $332-$348번지의 메모리를 보조 메모리의 $800번지 이후로 옮긴 다음 XFER 서브루틴을 사용해서 보조 메모리의 $800번지로 제어를 옮긴다. 그리고는 메인 메모리로 돌아와서 프로그램의 수행을 마친다.

이제 어느 정도 XFER 서브루틴에 대해 알았으면 이 루틴에 대해 한번 생각해 보자. 이것은 앞에서도 말했지만 메인 메모리와 보조 메모리간의 프로그램 수행을 바꾼다고 했다. 그런데 보통은 보조 메모리에는 아무 프로그램도 없으므로 이 서브루틴은 MOVEAUX 서브루틴과 같이 쓰이는 예가 많다. 그것은 앞의 프로그램만 보아도 그렇다. 마지막으로 이 서브루틴을 정리해 보면 다음과 같다.

(3) XFER 서브루틴의 요약

XFER 서브루틴($C314)

기　　능 : 메인/보조 메모리간의 프로그램 제어를 옮긴다.

사 용 법 : 파라메터를 세트하고 JMP $C314와 같이 한다(단 여기서는 JSR이 아니고 JMP이다)

레지스터 : A, X, Y 레지스터 모두 보존된다.

파라메터 : 캐리(C)
- 세트(C=1) : 보조 메모리로 제어가 옮겨간다.
- 리셋(C=0) : 메인 메모리로 제어가 옮겨간다.

오버플로우(V)
- 세트(V=1) : 보조 메모리의 제로페이지, 스택 상부 16K($DOOO-$FFFF) 를 사용한다.
- 리셋(V=0) : 메인 메모리의 것을 사용한다.

$3ED, $3EE번지 : 제어를 옮길 곳의 시작 번지를 가진다.

§4. 메모리 관리와 관계된 소프트 스위치

그러면 이번에는 메모리와 관계된 소프트 스위치를 직접 사용해 보기로 하자. 앞에서도 언급한 바가 있지만 소프트 스위치를 직접 작동시킨다는 것은 쉽지 않다. 게다가 실수로 보조 메모리에 스택이나 제로페이지 등이 세트되지 않은 상태에서 이 소프트 스위치를 건드려서 기계가 다운되는 일이 종종

있곤 한다. 그래서 이 부분은 사용에 각별한 주의를 요한다.

1. 16K 램카드

(1) 소프트 스위치의 이해

우선 Ⅱ+에도 있는 16K 램카드의 소프트 스위치부터 알아보자. 이것은 다른 곳에서도 소개된 일이 있으나, 앞으로 소프트 스위치에 대한 이해를 위해서는 이미 알고 있는 것부터 설명하는 것이 빠르기 때문이다. 사실 보조 메모리를 사용하는 소프트 스위치나 16K 소프트 스위치나 그 사용 방법은 거의 같다.

다음 그림을 보자. 이 그림은 상위 16K의 RAM/ROM의 메모리 맵이다.

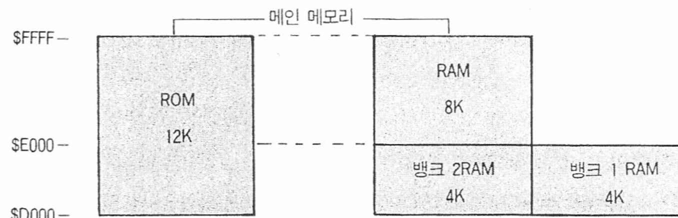


그림 4. 5 상위 16K의 메모리 맵

이 그림에서 ROM과 RAM이 같은 메모리 번지를 가지고 있다는 것과 RAM중 $D000-$DFFF부분은 두 개의 뱅크로 되어있다는 것을 알 수 있다. 그런데 CPU는 이들을 한 번에 어드레싱할 수 없기 때문에 이들 중 어느 하나만을 사용해야 하며 이를 위한 스위치 구실을 하는 것이 필요하다. 단 이것은 보통 하드웨어적인 스위치가 아니라 소프트웨어적으로 움직이는 것이므로 소프트 스위치라고 부른다.

RAM이나 ROM을 선택할 때에는 간단히 "램 선택", "롬 선택"으로 할 수도 있겠지만, 더 실용적인 사용을 위해 다음과 같은 네 가지 사용법을 만들어 놓았다.

① 롬 읽고 롬 쓰기(read/write with ROM)

② 롬 읽고 램 쓰기(read ROM, write RAM)

③ 램 읽고 롬 쓰기(read RAM, write ROM)

④ 램 읽고 램 쓰기(read/write with RAM)

여기서 롬 읽고 롬 쓰기라는 말은 CPU가 메모리를 억세스할 때 LDA명령 등으로 메모리의 데이타를 읽을 때에는 롬으로, STA 명령 등으로 메모리에 쓸 때에도 롬으로 한다는 것이다. 그런데 여기서 한 가지 짚고 넘어갈 것이 있다. 과연 롬에 데이타를 쓸 수 있는가 하는 것이다. 독자 여러분도 잘 알다시피 롬은 오직 읽는 것만 가능하기 때문이다. 그렇기 때문에 이 글을 읽는 데 혼란이 있을 수도 있겠는데 이는 램에 데이타를 쓰지 않는, 다시 말하면 램에 이미 있는 데이타를 보존하는 것으로 생각하

면 된다.

앞서서 이 부분은 기존의 Ⅱ+와 같다고 했는데, 기능적으로는 같지만 몇 개의 소프트 스위치가 추가되었다. 즉 전에는 단순히 램 또는 롬에 읽고 쓰고 하는 상태를 바꾸는 일만을 하는 소프트 스위치만 있었는데, Ⅱe에는 현재 램과 롬 어느 쪽이 작동되고 있나를 확인하는 소프트 스위치가 추가되었다는 점이다. 이것은 비단 이 16K램뿐만이 아니라 보조 메모리에 관한 소프트 스위치에도 이 상태 확인 스위치는 있다는 것이다.

(2) 16K 램카드와 관련된 소프트 스위치

이제 구체적으로 설명해 보기로 하자. 여기서 말하는 "램"이란 16K 램카드($D000-$FFFF)이다. 또한 "뱅크"란 $D000-$DFFF의 영역을 말한다(그림 4. 5 참고). 또한 "번지를 읽는다", "쓴다"라는 말은 그 소프트 스위치에 해당하는 번지를 PEEK/POKE(또는 LDA/STA)한다는 뜻임을 이 자리에서 밝혀둔다.

$C080(= 49280 = -16256)

이 번지를 읽거나 쓰면 현재의 상태는 다음과 같다.

- 램에서는 읽는 것만이 가능하며 쓰는 것은 불가능하다.
  램 뱅크($D000-$DFFF)는 두번째 뱅크가 사용된다.

$C081(= 49281 = -16255)

이 번지를 연속 두 번 읽으면 다음과 같이 된다.

- $D000-$FFFF의 램에는 데이타를 쓸 수 있다.
  읽는 것은 롬에서 한다.
  두번째 뱅크가 사용된다.
  (처음 애플Ⅱe 또는 Ⅱ+의 전원을 켜면 이 상태로 된다.)

$C082(= 49282 = -16254)

이 번지를 읽으면 다음과 같이 된다.

- 오직 롬에서만 읽고 쓰는 것이 가능하다.
  램의 상태는 변하지 않는다.
  두번째 뱅크가 사용된다.

$C083(= 49283 = -16253)

이 번지를 연속 두 번 읽으면 상태는 다음처럼 된다.

- 읽고 쓰는 것은 램에서 이루어진다.
  CPU는 롬에 전혀 관여하지 않는다.
  두번째 뱅크가 사용된다.

$C088(= 49288 = -16248)

이 번지를 읽거나 쓰면 현재의 상태는 다음과 같다.

– 램에서는 읽는 것만이 가능하며 쓰는 것은 불가능하다.
  램 뱅크($D000-$DFFF)는 첫번째 뱅크가 사용된다.

$C089(= 49289 = -16247)

이 번지를 연속 두 번 읽으면 다음과 같이 된다.

– $D000-$FFFF의 램에는 데이타를 쓸 수 있다.
  읽는 것은 롬에서 한다.
  첫번째 뱅크가 사용된다.

$C08A(= 49290 = -16246)

이 번지를 읽으면 다음과 같이 된다.

– 오직 롬에서만 읽고 쓰는 것이 가능하다.
  램의 상태는 변하지 않는다.
  첫번째 뱅크가 사용된다.

$C08B(= 49291 = -16245)

이 번지를 연속 두 번 읽으면 상태는 다음처럼 된다.

– 읽고 쓰는 것은 램에서 이루어진다.
  CPU는 롬에 전혀 관여하지 않는다.
  첫번째 뱅크가 사용된다.

다음은 Ⅱe에서 추가된 부분이다.

$C011(= 49169 = -16367)

이 번지의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

– 최상위 비트 = 0 (메모리값〈128) : 뱅크 1이 사용되고 있다.
  최상위 비트 = 1 (메모리값〉127) : 뱅크 2가 사용되고 있다.
  (전원을 켠 직후에는 뱅크 2가 사용된다.)

$C012(= 49170 = -16366)

이 번지의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

– 최상위 비트 = 0 (메모리값〈128) : 롬에서 읽는다.
  최상위 비트 = 1 (메모리값〉127) : 램에서 읽는다.
  (전원을 켠 직후에는 롬에서 읽는다.)

(3) 소프트 스위치 사용시 주의 사항

우리가 흔히 롬에 있는 베이직 인터프리터를 램으로 옮길 때에는 다음과 같이 입력한다.

```
]CALL-151
* C081 N D000,D000<FFFFM N C083
```

의미는 다음과 같다. 먼저 C081이라고 한 것은 현재의 상태를 롬에서 읽고 쓰게 만든다. 그 다음의 것은 $D000부터 $FFFF까지의 메모리를 옮기는 명령인데, 원래 번지와 옮겨지는 번지가 같기 때문에 같은 번지에서 읽고 쓰게 된다. 그런데 현재의 상태가 롬 읽고 램 쓰는 상태이므로 롬에서 읽고 램에 쓰게 된다. 그리고 마지막의 C083은 현재의 상태를 램 읽고 램 쓰기로 바꾸는데, 이는 램으로 옮겨진 베이직 인터프리터와 모니터를 쓰기 위해서이다.

여기서 한 가지 조심할 것이 있다. 다음은 소프트 스위치를 쓸 때에는 조심해야 한다는 것을 말해주는 한 예이다. 만일 램카드의 $E003번지에 있는 값을 프린트할 때 다음과 같은 프로그램을 만들었다고 하자.

```
300-  AD 83 C0    LDA $C083    ;램 읽기 모드
303-  AD 83 C0    LDA $C083    ;(두 번 읽는다.)
306-  AD 03 E0    LDA $E003    ;$E003번지의 값을 읽음.
309-  20 DA FD    JSR $FDDA    ;이 값을 프린트.
30C-  AD 81 C0    LDA $C081    ;원래 모드로 돌아온다.
30F-  AD 81 C0    LDA $C081    ;(두 번 읽는다.)
312-  60          RTS
```

이렇게 하면 될 것 같지만 실은 이렇게 되지 않는다. 아마 기계가 정지한 상태로 계속 있을 것이다. Ⅱe의 경우는 리셋을 누르면 정지되기는 하나 Ⅱ+의 경우에는 그것조차 되지 않을 것이다.

이유는 간단하다. $C083을 읽음으로써 램 상태로 해서 $E003번지를 읽는 것까지는 좋았다. 그러나 다음 $FDDA로 점프한 것이 잘못이다. 이것은 롬에 있는 서브루틴이지 램에 있는 것은 아니기 때문이다. 그렇기 때문에 그곳에서 프로그램이 정지해 버린 것이다.

그럼 Ⅱ+에서는 왜 리셋이 걸리지 않는가? $FDDA로 점프했을 때 그곳에는 00이 들어있기 때문에 브레이크가 걸리게 되나, 브레이크를 처리해 주는 서브루틴 또한 롬에 있기 때문에 램에서 뱅뱅 돌게 된다. 리셋키를 누르면 리셋 벡터를 읽어 그 값으로 점프하는데, 그것 또한 롬에 있다.

그렇기 때문에 다음과 같이 프로그램해야 한다.

```
300- AD 83 C0    LDA $C083    ;램 읽기 모드.
303- AD 83 C0    LDA $C083    ;(두 번 읽는다.)
306- AD 03 E0    LDA $C003    ;$E003번지의 값을 읽음.
309- 8D 81 C0    STA $C081    ;롬 읽기 모드.
30C- 8D 81 C0    STA $C081    ;(두 번 읽는다.)
30E- 20 DA FD    JSR $FDDA    ;어큐뮬레이터의 값을 프린트.
312- 60          RTS
```

앞으로 소프트 스위치를 사용할 때는 이 점에 주의해서 사용하기 바란다.

2. $200-$BFFF까지의 메인 / 보조 메모리 관리 소프트 스위치

여기서는 보조 메모리를 사용하는 소프트 스위치 중 $200-$BFFF까지를 관리하는 소프트 스위치를 알아보기로 하자. 여기서 64K 메모리 전부가 아니라 이렇게 부분만을 관리하는 소프트 스위치를 만든 것은 $0~$1FF번지가 제로페이지와 스택이고 $D000~$FFFF는 롬과 함께 세 개(롬, 메인 램, 보조 램)로 되어 있어서 앞의 것과는 그 의미가 약간 다르기 때문이다.

특히 이 보조 메모리에 관한 소프트 스위치는 한 개의 소프트 스위치가 다른 용도로 사용된다는 것을 알아두어야 한다. 예를 들면 $C000번지는 읽을 때는 현재 어느 키가 눌렸느냐를 알아보는 것이지만 쓸 때에는 80STOREC라는 새로운 기능을 한다는 것이다. 게다가 기존의 Ⅱ+의 소프트 스위치는 읽거나 쓰는 것 어느 것을 행해도 스위치 동작을 했는데, 이제부터 설명하는 것은 읽기 또는 쓰기의 어느 하나만을 행했을 때에만 동작한다는 사실이다(물론 다 그런 것은 아니다). 그러면 하나씩 구체적으로 설명하기로 하자.

$C002(=49154=-16382)

RAMRDM - 이 소프트 스위치에 어떤 값이든지 쓰면(읽으면 안된다) CPU는 메인 메모리의 $200-$BFFF번지에서 읽게 된다. 전원을 켠 직후의 상태이다 (뒤에서 설명할 80STORE가 제거되어 있어야 한다).

$C003(=49155=-16381)

RAMRDA - 이 소프트 스위치에 어떤 값이든지 쓰면(읽으면 안된다) CPU는 보조 메모리의 $200-$BFFF번지에서 읽게 된다(80STORE가 제거되어 있어야 한다).

$C004(=49156=-16380)

RAMWRTM - 이 소프트 스위치에 어떤 값이든지 쓰면(읽으면 안된다) CPU는 메인 메모리의 $200-$BFFF번지에서 쓰게 된다(80STORE가 제거되어 있어야 한다).

$C005(=49157=-16379)

RAMWRTA – 이 소프트 스위치에 어떤 값이든지 쓰면(읽으면 안된다) CPU는 보조 메모리의 $200-$BFFF 번지에서 쓰게 된다(80STORE가 제거되어 있어야 한다).

그런데 이 네 개의 소프트 스위치를 사용하는 데에는 왜 먼저 80STORE를 세트해야 할까? $0~$BFFF까지의 48K램은 비디오램을 포함하고 있다. 그렇기 때문에 메인/보조 메모리의 읽기/쓰기 전환은 당연히 비디오램도 바꾸어 버려서 현재 화면이 망가지는 경우 등이 가끔 일어나곤 한다. 그렇기 때문에 이 80STORE라는 것을 둔 것이다. 만일 이것이 세트된 상태라면 위의 네 개 소프트 스위치는 무기력하게 된다. 80STORE라는 이름이 의미하는 대로 이것은 80컬럼 화면을 위해서 만들어진 것이다.

이 소프트 스위치를 적절히 사용하면 다음과 같은 트릭을 사용할 수도 있다. 이는 $200~$BFFF까지의 메인 메모리를 간단히 보조 메모리로 옮겨 버릴 것이다.

```
]CALL-151
* C002:00 N C005:00 N 200<200. BFFFM N C004:00
*
```

처음에 $C002:00 $C005:00으로 메인 메모리에서 읽고 보조 메모리에서 쓰는 상태로 만든다. 다음 200<200. BFFFM은 200번지에서부터 BFFF번지까지의 것을 읽고 쓰는데, 메인에서 읽고 보조 메모리에 쓰게 된다. 이것은 위 16K램카드에서 롬의 베이직을 램으로 옮기는 것과 같은 부류라고 생각하면 된다. 마지막의 $C004:00은 보조 메모리로 쓰게 되어있는 것을 메인 메모리에 쓰도록 원상복귀시킨다. 이렇게 하지 않으면 보조 메모리의 비디오 램에 화면 표시를 하므로 화면이 보이지 않게 되기 때문이다.

물론 이 네 개의 소프트 스위치도 그 상태를 확인해 보는 것이 하나씩 있다. 다음을 보자.

$C013(=49171=-16365)

RDRAMRD–이 번지의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0 (데이타값<128) : 메인 메모리에서 읽는 상태.

최상위 비트=1 (데이타값>127) : 보조 메모리에서 읽는 상태.

관계되는 소프트 스위치 : $C002(RAMRDM), $C003(RAMRDA)

$C014(=49172=-16364)

RDRAMWRT–이 번지의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0 (데이타값<128) : 메인 메모리로 쓰는 상태.

최상위 비트=1 (데이타값>127) : 보조 메모리로 쓰는 상태.

관계되는 소프트 스위치 : $C004(RAMWRTM), $C005(RAMWRTA)

3. 제로페이지와 스택 및 상부 16K($D000－$FFF)를 위한 메인 / 보조 메모리 관리 소프트 스위치

어떤 독자들은 왜 제로페이지나 스택 등이 이전의 48K 영역과 같은 소프트 스위칭을 하지 않는가에 대해 의문을 가진다. 그러나 128K 프로그래밍을 많이 해 본 독자라면 이렇게 하는 편이 훨씬 편하다는 것을 알게 될 것이다.

대부분의 경우 $200－$BFFF번지를 보조 메모리의 것으로 전환했다 하더라도 제로페이지와 스택은 현재의 것을 그대로 사용하는 것이 프로그래밍에 편리하다. 또한 $D000－$FFFF영역은 램만 두 가지(메인/보조)가 있는 것이 아니라 롬도 위치하고 있다. 그렇기 때문에 이렇게 별도의 소프트 스위치를 만든 것이다.

본 소프트 스위치를 사용하는 것은 더 조심스럽게 해야 한다. 왜냐하면 제로페이지나 스택은 CPU나 모니터 루틴, 베이직 인터프리터 등에서 유용하게 쓰이기 때문에 이를 임의로 바꾸었을 때에는 예기치 않는 결과를 초래할 수 있기 때문이다. 다음을 보자.

$C008(＝49160＝－16367)

ALZPM－이 메모리 번지에 데이타를 쓰면 제로페이지, 스택, 상부 16K($D000－$FFFF)가 메인 메모리의 것으로 된다.

$C009(＝49161＝－16366)

ALZPA－이 메모리 번지에 데이타를 쓰면 제로페이지, 스택, 상부 16K($D000－$FFF)가 보조 메모리의 것으로 된다.

$C016(＝49174＝－16362)

RDALZP－이 번지의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트＝0 (데이타값<128) : 메인 메모리의 것이 선택된 상태.

최상위 비트＝1 (데이타값>127) : 보조 메모리의 것이 선택된 상태.

관계되는 소프트 스위치 : $C008(ALTZPM), $C009(ALTZPA)

이 소프트 스위치는 앞의 48K 부분과는 달리 읽기/쓰기가 따로 나뉘어져 있는 것이 아니라 위와 같이 하나의 소프트 스위치로 읽기/쓰기를 동시에 전환하게 된다.

4. 메모리 전환 관계 제로페이지

아래의 제로페이지 번지는 앞의 MOVEAUX 서브루틴을 설명할 때 한번 언급한 일이 있을 것이다.

그런데 이 서브루틴은 MOVEAUX 외에 기존의 MOVE 서브루틴에도 동일한 형식으로 사용한다.

참고 MOVE 서브루틴은 모니터의 M명령과도 상통하는 것으로, 메인/보조 어느 곳에서든지 메모리를 이동시킨다. 단 같은 메모리(메인 또는 보조)상태에서만 움직인다. 이를 잘 쓰면 MOVEAUX와 같은 일을 할 수 있게 만들 수도 있다. 절대 번지는 $FE2C(65068)이다.

그럼 아래를 보자

$3C, $3D (60, 61)–A1 : 옮길 프로그램의 시작 번지를 담는다.

$3E, $3F (60, 61)–A2 : 옮길 프로그램의 끝번지를 담는다.

$42, $43 (60, 61)–A4 : 옮겨질 곳의 시작 번지를 담는다.

이 세 개는 모두 하위 바이트 먼저 들어가게 된다.(A1H, A1L)

5. 비디오 상태 전환과 관계 있는 소프트 스위치

사실 이것은 다음의 더블 그래픽을 설명할 때 같이 할 수도 있다. 그러나 메모리와 관계되는 부분이고 이 장에서 서술하지 않으면 안되기 때문에 여기서 서술하기로 한다.

비디오램 부분이 텍스트/GR·영역과 HGR 영역의 두 개로 되어있다는 것은 독자들은 이미 알고 있을 것이다. 그런데 이것은 우리가 알고 있는 메인 메모리 이외에 보조 메모리에도 있다(단 보조 메모리에는 2페이지가 없다). 이것은 다음과 같은 소프트 스위치를 씀으로써 전환할 수 있으나, 이들 소프트 스위치는 다른 것에 비해 사용법이 복잡하다. 먼저 아래의 두 소프트 스위치를 보자. 이것은 앞에서도 잠깐 설명한 80STORE라는 것으로, 앞의 48K 선택 때에도 이 스위치가 제거되어 있어야만 메인/보조 48K가 선택이 가능하다고 했다. 그러나 이 80STORE는 뒤에 나올 소프트 스위치로 그 기능을 바꾼다. 게다가 이 80STORE라는 스위치 자체도 읽을 때에는 바로 직전에 눌려진 키를 체크하는 것이 된다는 것이다.

아래에 서술하는 것을 잘 보기 바란다.

$C000(=49152=-16384)

80STOREC–이 번지에 어떤 값을 써넣으면 80STORE 기능을 제거한다.

$C001(=49153=-16383)

80STORES–이 번지에 어떤 값을 써넣으면 80STORE 기능을 세트한다.

$C018(=49176=-16360)

RD80STORE–이 번지의 최상위 비트(MSB)는 다음과 같은 뜻을 가진다.

최상위 비트 =0 (데이타값<128) : 80STORE가 제거되었음.

최상위 비트 =1 (데이타값>127) : 80STORE가 세트되었음.

관계있는 소프트 스위치 : $C000 (80STOREC), $C001 (80STORES)

일단 한번 80STORE 기능이 세트되면 뒤에 소개할 PAGE2 소프트 스위치가 텍스트/GR 1페이지와 HGR 1페이지가 메인 메모리냐 보조 메모리이냐를 결정하게 된다. 그렇기 때문에 XFER 서브루틴 등으로 보조 메모리의 프로그램으로 수행을 넘긴 경우라면, 바로 이 소프트 스위치를 사용하게 될 때 보조 메모리의 텍스트 화면 및 그래픽 화면을 사용하므로 화면이 안보이는 일이 없다. 128K를 두 대의 64K 애플처럼 사용하는 기능은 바로 이들 소프트 스위치를 적절히 사용하여 만들어낸 것이다.

그러면 이번에는 그 PAGE2라는 소프트 스위치에 대해 알아보기로 하자. 다음을 보라.

$C054(=49236=−16300)

PAGE2C—이 소프트 스위치를 읽거나 쓰게 되면 PAGE2 기능을 제거한다. 만일에 80STORE 기능이 제거된 채로 이 소프트 스위치를 움직이게 되면 기존의 Ⅱ+와 같이 그래픽 1페이지(TEXT/GR은 $400-$7FF, HGR은 $2000−$3FFF)로 전환하게 되고, 80STORE 기능이 세트된 채로 이 소프트 스위치를 사용하면 텍스트 및 GR/HGR의 1페이지를 메인 메모리의 것으로 사용하게 한다.

$C055(=49237=−16299)

PAGE2S—이 소프트 스위치를 읽거나 쓰게 되면 PAGE2 기능을 설정한다. 만일에 80STORE 기능이 제거된 채로 이 소프트 스위치를 움직이게 되면 기존의 Ⅱ+와 같이 그래픽 2페이지(TEXT/GR은 $800−$BFF, HGR은 $4000−$5FFF)로 전환하게 되고, 80STORE 기능이 세트된 채로 이 소프트 스위치를 사용하면 텍스트 및 GR/HGR의 1페이지를 보조 메모리의 것으로 사용하게 한다.

$C01C(=49180=−16356)

RDPAGE2—이 소프트 스위치의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0 (데이타값<128): PAGE2 기능이 제거된 상태.

최상위 비트=1 (데이타값>127): PAGE2 기능이 설정된 상태.

관계된 소프트 스위치: $C054 (PAGE2C), $C055(PAGE2S)

다음 그림을 보면 이 스위치에 대한 개념을 확실히 이해할 것이다.

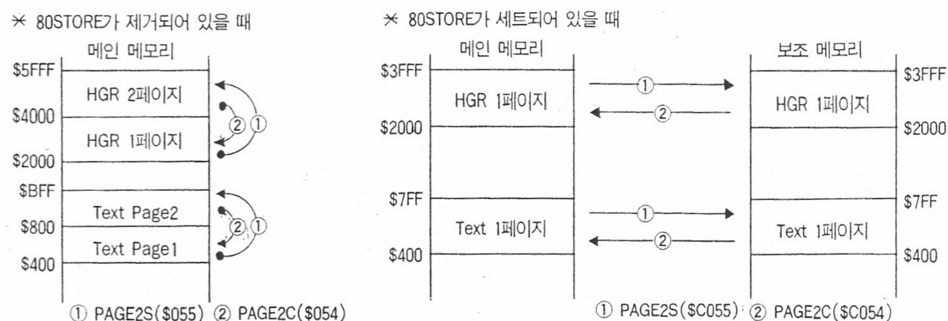


그림 4. 6 $C4054와 $C055 소프트 스위치의 동작

이 정도로 128K 메모리 사용에 관한 소프트 스위치의 소개를 마치겠다. 사실 여기 소개한 소프트 스위치가 128K를 64K씩 제어하는 모든 소프트 스위치이다. 특히 여기 있는 예제는 반드시 해보는 것이 이 128K를 이해하는 데 좋을 것이다.

§5. 128K를 능가하는 애플Ⅱe

아직까지는 64K의 Ⅱ+가 판을 치고 있는 우리나라의 사정으로 비추어 볼 때 애플Ⅱe의 128K도 매우 큰 용량이라고 생각된다. 64K를 사용할 때에 발생했던 메모리 부족 등의 문제도 128K가 부족한 경우까지 발전되는 일은 거의 없다. 사실 64K가 부족할 정도의 프로그램을 짤 수 있는 실력을 가진 독자도 별로 없을 것이다.

그런데 이것은 우리나라 이야기이고 애플의 본고장인 미국의 이야기는 조금 다르다. 그들은 지금 128K도 부족해서 256K, 512K, 1메가로 확장시키는 인터페이스 카드를 사용하고 있으며 최근에는 16메가바이트까지 확장시키는 카드도 만든다고 한다. 물론 미국에서는 애플웍스 등 많은 데이타를 필요로 하는 프로그램이 있기 때문이지만, 조만간 우리나라에서도 이런 프로그램이 나오지 않으리라는 법은 없기 때문에 이들 카드의 생산 또는 시판은 필요하다고 본다.

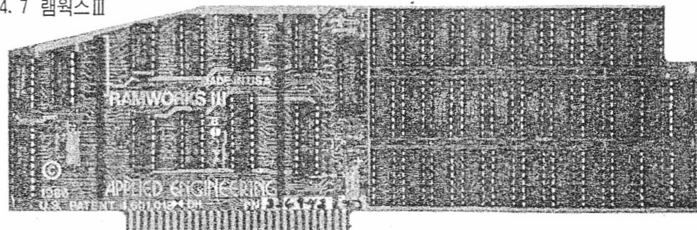
1. 애플의 각종 확장 램 카드

이 장에서는 미국에서 판매되고 있는 애플 확장 램 카드들에 대해 간단히 알아보고 넘어가기로 하자.

(1) 어플라이드 엔지니어링(AE)의 램웍스 Ⅲ(RamWorks Ⅲ)

이 제품은 64K 버전에서부터 16메가바이트 버전까지의 다양한 제품이 나와 있다. 가격은 512K가 2백69달러, 1메가바이트가 3백69달러이다. 모든 애플웍스(AppleWorks) 프로그램에서 오토 로드(auto load)가 가능하며 프린터 버퍼를 자체에 내장하고 있다. 또한 RGB와 16비트 65C816 CPU를 접속할 수 있는 포트가 내장되어 있다. 확장 램 포트도 내장되어 있다. 이 제품은 Ⅱe, Ⅱc, Ⅱ$_{GS}$버전이 있는데

그림 4. 7 램웍스Ⅲ

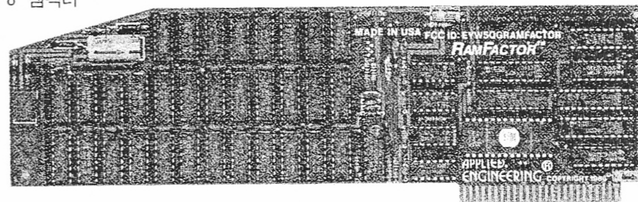


Ⅱc 버전은 Z-RAM Ultra, Ⅱ$_{GS}$버전은 gsRAM이라는 명칭으로 판매하고 있다. 물론 서로간에는 다소의 기능차가 있다.

(2) 어플라이드 엔지니어링(AE)의 램팩터(RamFactor)

앞에서 소개한 제품이 Ⅱe의 보조 슬롯을 사용한 것이라면 이 제품은 애플의 보통 1~7번 슬롯을 사용하는 제품이다. 그렇기 때문에 Ⅱ+에서도 사용이 가능하다.그 기능은 앞의 램웍스 Ⅲ과 거의 같으나, 가격이 약간 비싸다. 1메가바이트 버전이 3백89달러이고 여기에 65C816 16비트 CPU 카드를 추가할 수 있는데, 이 카드는 1백59달러이다.

그림 4. 8 램팩터



(3) 체크메이트 테크놀로지(Checkmate technology)의 멀티램(MultiRam)

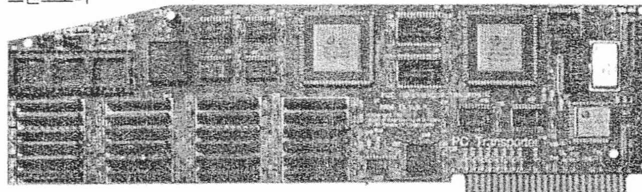
이 카드는 다른 것과는 달리 램카드 내에 RGB 카드가 내장되어 있다(그렇지 않은 멀티램도 있다). 가격 또한 앞의 램웍스나 램팩터에 비하여 좀 저렴한 편이다. 이것은 64K버전에서 1.5메가 버전까지의 제품을 구비하고 있다. 가격은 1메가 버전에 RGB카드가 내장된 형태가 2백79달러이다.

2. PC 트랜스포터 카드

1983년에 발표한 IBM PC는 IBM이라는 이름 석 자만 가지고 현재 전세계 PC시장을 상당 부분 잠식하고 있다. 국내의 PC 사정도 이와 같아서 IBM호환기의 시장 점유율은 점점 늘어만 간다. 게다가 MSX의 보급률도 점점 늘어, 한때에는 한국의 PC시장을 형성했고 지배해 왔던 애플의 설 땅이 점점 좁아지고 있다.

IBM의 위세가 대단하게 되자 애플의 많은 하드웨어 기술자들은 한 데 모여서 애플에서 IBM의 소

그림 4. 9 PC 트랜스포터



프트웨어를 구동시킬 수 있는 하드웨어를 설계하기에 이르렀고, 마침내 어플라이드 엔지니어링에서 "PC 트랜스포터(PC Transporter)"라는 이름의 IBM 에뮬레이팅 카드가 발매되기에 이르렀다. 가격은 1988년 현재 IBM 640K 모드에서 6백99달러, 360K IBM 포맷 5.25" 디스크 드라이브가 3백99달러이다.

이 카드에는 16비트 데이타 버스를 가진 NEC의 V30 CPU가 내장되어 있다. IBM-PC/XT에 내장된 인텔의 CPU 8088은 4.77MHz의 클럭 속도를 가지며 8비트의 데이타 포트를 가진다. 이렇게 볼 때 PC 트랜스포터 카드는 표준 XT보다도 더 빠른 속도를 가지게 된다. 게다가 여기에는 연산 전용의 Co-Processor인 8087-2칩을 끼울 소켓을 내장하고 있어서 이미 그 기능으로도 XT를 능가한다.

이 카드의 설계에는 당시 애플의 쟁쟁한 엔지니어들이 거의 모두 참여했다. 애플사 초기에 워즈니악을 도와 디스크 컨트롤러를 만든 클리프 휴스턴, 애플 ⅡGS에서 3.5인치 디스크를 컨트롤하는 IWM(Intergreated Woz Machine)칩을 설계하고 이전에 애플Ⅲ의 디자인 총책임자이기도 한 벤델 샌더, 애플 Ⅱe와 Ⅱc의 엔지니어링 프로젝트 메니저인 피터 퀸, 애플의 모뎀 및 로컬 네트워크의 설계자인 봅 래슬리 등이 이 카드 설계에 참여했다. 최대 768K의 메모리를 사용하여 128K를 자체 시스템이 사용하게 하고 나머지 640K를 IBM PC 에뮬레이팅 모드에서 사용한다고 한다. 이것을 애플웍스에서 쓸 때에는 750K의 데스크탑 메모리를 그레이드 업시킬 수 있다.

카드에 있는 VLSI에는 IBM-PC에 대한 I/O로 6502가 에뮬레이트할 수 있는 모든 하드웨어 포트가 내장되어 있다. PC 트랜스포터는 6502가 제공하는 데이타에 대하여 언제나 자체의 CPU에 조회하게 되는데, 이 VLSI를 사용하여 애플이 표준 IBM 칩이나 8237A DMA 컨트롤러, 8259A 인터럽트 컨트롤러, 8255A PPI, 게임 컨트롤러, 프린터를 위한 병렬 인터페이스, 모뎀에서 이용되는 비동기식 통신 인터페이스, ProDOS나 MS-DOS로 포매팅된 하드디스크 장치 등의 기능을 에뮬레이트하게 된다. 작동하는 방법은 먼저 프로도스를 불러 PC 트랜스포터 프로그램을 가동시킨 다음 메뉴 방식의 프로그램으로 시스템을 규정하는 방식으로 이루어진다.

모니터는 기본적으로 CGA와 애플 ⅡGS방식의 아날로그 RGB 모니터 중에서 선택하게 되어있으나 EGA까지도 확장이 되며, IWM칩을 이용해서 애플ⅡGS, 매킨토시 스타일의 3.5인치 드라이브와 IBM 스타일의 3.5인치 디스크 드라이브 중 어느 것이나 자유롭게 쓸 수 있다. 기존 IBM 스타일의 디스크 드라이브를 사용하기 위해서는 기존 34핀의 것을 애플용인 19핀 방식(DB-19)으로 바꾸어 연결해야 하는데, 이것을 하드웨어에 대한 지식이 없는 사람이 하기에는 조금 무리이므로 어플라이드 엔지니어링사에서는 아예 19핀으로 개조한 IBM 스타일의 드라이브를 판매한다고 한다.

이상으로 PC 트랜스포터에 대한 설명을 마친다. 사실 애플 컴퓨터는 나온 지 12년이 넘은 구형이다. 그런데도 아직까지 살아남은 것은 수많은 소프트웨어의 장벽과 함께 많은 확장 카드가 그 기계의 성능을 증가시키고 있기 때문이다. IBM 에뮬레이션 카드까지 발매된 지금, 애플의 수명은 언제까지일지 예측할 수 없게 되었다.

제 5 장

Ⅱe의 그래픽과 사운드

§1. 80컬럼 텍스트 화면
§2. 더블 고해상도 그래픽
§3. 더블 고해상도 그래픽의 실제 사용
§4. Ⅱe의 대표적인 그래픽 프로그램
§5. Ⅱe의 사운드 기능
§6. 애플의 뮤직 소프트웨어

§1. 80컬럼 텍스트 화면

많은 사람들이 애플의 80컬럼 화면이 기존의 바이덱스 카드에 비해 달라진 것이 없다고들 한다. 하지만 만일 그렇다면 이렇게 장황하게 늘어 놓지도 않았을 것이다. 이 의문을 풀기 위해서는 다음을 보라.

바이덱스 카드와 Ⅱe의 80컬럼과의 차이점

① 바이덱스 카드는 별도의 확장 장치이므로 그래픽 화면과 같이 쓸 수 없다. 그러나 Ⅱe의 80컬럼 화면은 이것을 해결한다. 즉 HGR(또는 Double-HGR) 화면에서 화면 하단에 네 줄의 80컬럼 텍스트를 쓸 수 있다.

② 바이덱스 카드는 자체에 텍스트 화면을 위한 비디오램을 내장하고 있는데 Ⅱe의 80컬럼은 메인 램에 위치한다. 그렇기 때문에 POKE문을 이용한 각종 트릭이 가능하다.

③ 바이덱스 카드는 애플사의 공식 카드가 아니기 때문에 애플웍스 등 많은 프로그램이 이 카드에서 동작하지 않는다(바이덱스 카드용 애플웍스도 있지만 이것은 개조된 것이며, 대부분의 프로그램들은 개조되지 않았다).

1. 메모리 구성

그러면 80컬럼을 한번 파헤쳐 보기로 하자. 80컬럼의 메모리 구성은 제2장에서 소개한 표를 보기 바란다. 80컬럼은 메인 메모리의 $400-$7FF번지와 보조 메모리의 $400-$7FF번지의 메모리를 사용한다.

보통 베이직에서는 잘 사용되지 않지만 텍스트(기존의 40컬럼을 말한다) 페이지는 $400-$7FF의 1페이지와 $800-$BFF의 2페이지의 두 개가 있다. 그러나 이 80컬럼 화면은 메인 메모리와 보조 메모리의 것을 합해서 만든 1페이지가 있고 2페이지는 존재하지 않는다. 더블 고해상도의 경우도 마찬가지로 2페이지가 없다.

이 80컬럼 메모리에 대해 자세히 설명하면 메인 메모리는 화면상의 홀수 컬럼을 담당하고 보조 메모리는 화면상의 짝수 컬럼을 담당한다. 즉 화면 최상단은 메인 메모리의 $400번지이고 그 다음 칸은 보조 메모리의 $400번지, 그 다음 칸은 메인 메모리의 $401번지 순으로 나간다는 것이다. 자세한 사항은 앞장을 참조하기 바란다.

여기서 메인 메모리와 보조 메모리가 교대로 화면상의 점에 위치한다는 것은 Ⅱe의 확장 그래픽 모드의 공통적인 특성이다. 즉 더블 저해상도 화면(80×48도트)이나 더블 고해상도 화면(560×192도트)도 이 점에서는 마찬가지이다. 2페이지가 없는 것 또한 마찬가지이다.

2. 80컬럼 펌웨어 및 문자 입력

80컬럼상에서의 문자 코드는 기존의 40컬럼 모드와는 다르다. 독자 여러분이 처음 80컬럼 모드(80컬럼 문자와 여기서 ESC-4를 눌러 만든 40컬럼 문자를 말한다)를 보았을 때에 커서가 Ⅱ+의 반짝이는 것이나 Ⅱe 40컬럼 모드(80컬럼 모드에서 ESC-4를 눌러 만든 40컬럼이 아닌 전원을 켠 직후의 40컬럼 상태를 말한다)의 체커보드 문자와는 다른 반전된 커서임을 보았을 것이다. 커서가 이렇게 된 이유는 80컬럼 상태에서는 플래시(FLASH) 문자가 없다는 것이다. 그 대신 인버스(INVERSE) 상태의 소문자가 존재한다는 것이다. 그렇기 때문에 소문자를 인버스시켰을 때에 나타나는 이상한 문자가 존재하지 않는다는 것이다. 이 기능은 실로 강력한 것으로 COPY Ⅱ PLUS 7.2를 실행해 보면 메뉴 선택에서 파일 이름이 소문자로 되어 있을 때에도 반전 막대 메뉴 선택에서 그 진가를 유감없이 나타낸다. 다음 그림은 기존 Ⅱ+(또는 Ⅱe의 40컬럼 모드)의 문자 세트와 80컬럼 모드의 문자 세트를 나타낸 것이다. 여기서 특수 문자라 하면 영문자를 제외한 모든 문자(단 [ ] } { 등은 제외)와 숫자를 나타낸 것이다.

16진 코드	10진 코드	40컬럼 모드 문자 세트	80컬럼 모드 문자 세트
$00~$1F	0~31	Inverse 대문자	Inverse 대문자
$20~$3F	32~63	Inverse 특수문자	Inverse 특수문자
$40~$5F	64 ~95	Flash 대문자	**(마우스 텍스트)
$60~$7F	96~127	Flash 특수문자	* Inverse 소문자
$80~$9F	128~159	***Normal 대문자	Normal 대문자
$A0~$BF	160~191	Normal 특수문자	Normal 특수문자
$C0~$DF	192~223	Normal 대문자	Normal 대문자
$E0~$FF	224~255	Normal 소문자	Normal 소문자

*Inverse 소문자가 Ⅱe의 80컬럼 모드에서 새로 생긴 부분이다.

**마우스 텍스트는 보통은 사용할 수 없고 롬 내에 마우스 텍스트가 들어있는 것에서 마우스 텍스트를 작동하는 제어 코드를 준 후에야 사용이 가능하다. 국내의 일부 Ⅱe는 이 롬을 장착하지 않은 것이 많으니 주의하기 바란다.

***$80−$9F의 부분은 화면에 이 코드를 POKE하여 나온다는 것이다. 원래 이 부분은 컨트롤 문자이니 주의하기 바란다.

표 5. 1 40 / 80컬럼 모드 문자 세트

처음 Ⅱe를 켠 후는 80컬럼 펌웨어는 사용되지 않은 상태이므로 40컬럼 모드 문자 세트로 되어있다. 그러나 80컬럼을 작동시키면 펌웨어가 시동되어 80컬럼 모드 문자 세트로 작동된다. 이것의 작동은 간단히 PR#3 또는 다음을 참조해서 하면 된다. 다음에 소개하는 것은 80컬럼과 관계있는 소프트 스위치이다.

$C000(=49152=−16384)

80STORES—이 번지에 어떤 값을 써넣으면 80컬럼 펌웨어가 작동(ON)된다.

대응되는 소프트 스위치 : $C001

상태 확인 스위치 : $C018 (RD80STORE)

$C001(=49153=−16383)

80STOREC—이 번지에 어떤 값을 써넣으면 80컬럼 펌웨어를 폐쇄(OFF)한다.

대응되는 소프트 스위치 : $C000

상태 확인 스위치 : $C018 (RD80STORE)

(이 80STORE 스위치는 또 다른 기능이 있다. 이 기능은 다음 더블 고해상도 그래픽을 설명할 때 소개하겠다.)

$C018 (=49176=−16360)

RD80STORE—이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0 (데이타값<128) : 80STORE 기능이 제거되었음.

최상위 비트=1 (데이타값>127) : 80STORE 기능이 세트되었음.

$C00C (=49164=−16372)

80COLC—이 번지에 어떤 값을 써넣으면 현재의 화면을 40컬럼 화면으로 만든다.

대응되는 소프트 스위치 : $C00D

상태 확인 스위치 : $C01F· (RD80C0L)

$C00D (=49165=−16371)

80COLS—이 번지에 어떤 값을 써넣으면 현재의 화면을 80컬럼 화면으로 만든다.

대응되는 소프트 스위치 : $C00C

상태 확인 스위치 : $C01F (RD80COL)

$C01F (=49183=−16353)

RD80C0L—이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0(데이타값<128) : 40컬럼 상태

최상위 비트=1(데이타값>127) : 80컬럼 상태

$C00E (=49166=−16370)

ALTCHARC—이 번지에 어떤 값을 써넣으면 40컬럼 모드 문자 세트(80컬럼이라도)를 사용한다.

대응되는 소프트 스위치 : $C00F

상태 확인 스위치 : $C01E (RDALTCHAR)

$C00F (=49167=−16369)

ALTCHARS—이 번지에 어떤 값을 써넣으면 80컬럼 모드 문자 세트를 사용한다.

대응되는 소프트 스위치 : $C00E

상태 확인 스위치 : $C01E (RDALTCHAR)

$C01E(=49182=-16354)
RDALTCHAR-이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.
최상위 비트=0 (데이타값〈128) : 40컬럼 모드 문자 세트 상태
최상위 비트=1 (데이타값〉127) : 80컬럼 모드 문자 세트 상태

다음에는 이들 펌웨어가 작동되었을 때의 문자 입력에 대해 알아보자. 일단 80컬럼 펌웨어가 작동되면 모니터의 KEYIN서브루틴이 작동하지 않는다. 따라서 이 루틴을 쓰는 프로그램은 80컬럼 모드에서는 작동하지 않는다. 그 대신 BINPUT($C8F6=-51446)에 의해 BASICIN($C305=-49925)이 사용된다.

*BINPUT, BASICIN : 이 루틴은 RDKEY가 만든 커서를 INVERT루틴에 의해 반전시킨 후 OURCH($57B=1403)와 OURCU($5FB=1531)가 계산한 화면상의 번지에 문자 코드를 기억시키고 GETKEY로 입력을 기다리게 한다. 펌웨어가 작동되지 않은 상태에서 CSW는 COUT1($FDF0=65008)을 지정한다. 그러나 일단 이것이 작동되면 CSW는 BASICOUT($C307=49927)을 지정하여 80컬럼 펌웨어 모드시의 출력 루틴을 부른다. 컨트롤키의 입력이 없으면 BPNCTL($C8CC=51404)이 OURCH($57B=1403), OURCU($5FB=1531)에 의해 계산한 화면상의 번지에 기억시키게 되고 문자 출력을 끝낸다.

*BPRINT($C8A1=51361)에 컨트롤 문자가 주어지면 CTL CHAR($CB99=52121)가 불려져서 VIDOUT와 같은 기능을 하게 된다.

3. 80컬럼의 기능

먼저 80컬럼의 시동은 베이직상에서는 PR#3이고 모니터상에는 *C300G(또는 *3 〈CTRL-P〉)를 입력하면 된다. 그리고 80컬럼 모드에서 다시 40컬럼 모드로 돌아올 때에는 〈ESC〉 키를 누른 뒤 〈CTRL-Q〉를 누르면 된다. PR#3으로 시작했으므로 PR#0으로 끝낼 것 같지만 이렇게 하면 80컬럼으로 되어있는 소프트 스위치나 제로페이지를 그대로 두기 때문에 문제가 생긴다.

80컬럼 모드에서는 NORMAL문자의 경우 기존의 것과 마찬가지로 대, 소문자 모두 사용이 가능하다. INVERSE의 경우에도 소문자 사용이 가능한데, 이는 이전에 대문자만 사용이 가능했던 것에 비하면 큰 발전이다. 단, 그 덕택(?)에 FLASH문자는 사용이 불가능하다.

80컬럼 모드에서는 베이직의 PRINT명령의 ,(커머)와 TAB, SPC 명령도 80컬럼에 맞게 동작한다(이 또한 바이덱스 카드와는 다른 점이다). 게다가 HTAB이 80컬럼까지 작동된다. 물론 32~35번지의 스크린 윈도우 또한 마찬가지이다.

앞에서 잠깐 언급한 바가 있지만 80컬럼 모드에서는 40컬럼을 사용할 수 있다고 했다(이것은 40컬럼 모드와는 다르다. 그 차이는 커서 모양을 보면 알 수 있다). 이것도 80컬럼 상태의 문자체 및 커서를 동일한 것을 사용한다. 또한 80컬럼 모드에서 ESC키를 누르면 화면은 반전된 "+"문자가 나타나서 ESC를 눌렀는지를 잘 알게 해준다.

40컬럼을 사용하려면 간단히 〈ESC〉 4를 누르면 된다.다시 80컬럼으로 돌아가려면 〈ESC〉 8을 누르면 된다. 80컬럼 상태에서 40컬럼으로 들어가면 화면의 오른쪽 절반은 화면에서 사라지게 된다. 이것은 80컬럼으로 돌아와도 나타나지 않는다.

4. 마우스 텍스트

80컬럼 모드에서 새로이 쓸 수 있는 문자 세트가 바로 이 마우스 텍스트이다. 이것은 기존의 Ⅱ+에서는 볼 수 없었던 것으로, Ⅱe에서 추가된 것이다. 그런데 마우스 텍스트는 주변기기인 마우스와는 전혀 관련이 없다는 것을 말해두는 바이다.

이 마우스 텍스트의 위력은 프로도스 시스템 디스켓 버전 2.0을 사용해보면 그 위력을 확실히 알 수 있다. 그런데 이것은 그냥 쓸 수 있는 것이 아니라 다음과 같은 준비 과정이 필요하다.

PRINT CHR$(27) : 마우스 텍스트를 가동시킨다.
INVERSE : 인버스 문자 상태로 만든다.

이렇게 하고 알파벳 A~Z과 @, [,], \, ^ 등의 문자를 입력해 보자. 화면에는 마우스 텍스트가 표시될 것이다. 이를 되돌릴 때에는 다음과 같이 한다.

NORMAL : 정상 문자 상태로 만든다.
PRINT CHR$(24) : 마우스 텍스트를 끈다.

PRINT CHR$(27)로 마우스 텍스트를 작동시키지 않은 상태에서는 INVERSE는 보통 문자 인버스 상태의 기능을 한다.

다음 프로그램을 한번 입력해 보라. 이것은 화면에 32개의 마우스 텍스트를 표시하게 될 것이다.

```
10 PRINT CHR$(4)"PR#3" : PRINT CHR$(27) ; CHR$(17)
20 PRINT CHR$(15) : PRINT CHR$(27)
30 FOR A=64 TO 95 : PRINT CHR$(A) ;: NEXT A
40 PRINT CHR$(24) ; CHR$(14) : END
```

다음 프로그램은 마우스 텍스트를 사용하는 재미있는 예이다. 실행예는 생략한다.

```
10 PRINT CHR$(4) "PR#3" : PRINT CHR$(27) ; CHR$(17)
20 PRINT CHR$(15) : PRINT CHR$(27)
30 FOR A=1 TO 39 : VTAB 10 : HTAB A : PRINT "FG";:NEXT A
40 PRINT CHR$(24) ; CHR$(14) : END
```

다음 그림은 32개의 마우스 텍스트 전체를 나타낸 표이다. 참고하기 바란다.

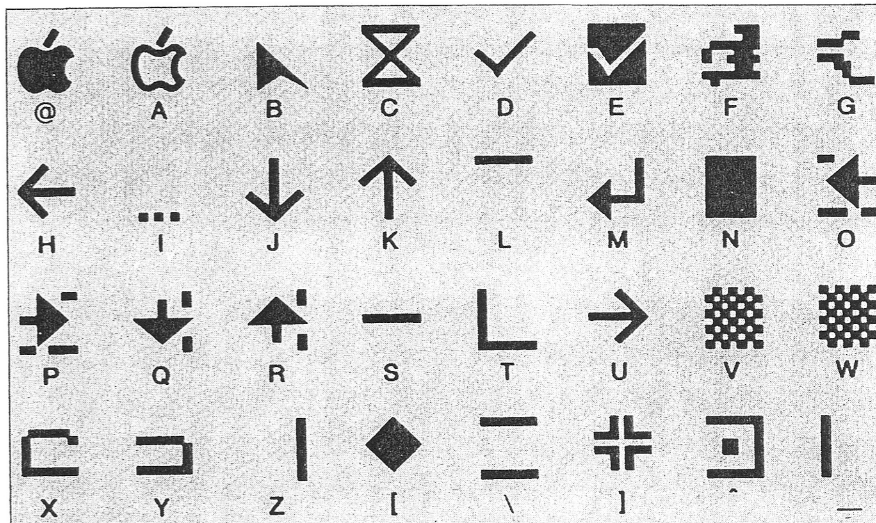


그림 5. 1 마우스 텍스트 코드

§ 2. 더블 고해상도 그래픽

개량형 애플Ⅱe의 하드웨어가 기존의 것에 비해 크게 향상된 부분이 바로 이 더블 고해상도 그래픽이라고 할 수 있다. 기존 애플의 해상도를 두 배로 늘림으로써 더욱 정밀한 그래픽 화면을 만들 수가 있고, 16색의 다양한 색상을 구사할 수 있다는 것은 애플 사용자에게는 참으로 획기적인 일일 것이다. 물론 지금은 이것보다 더 뛰어난 그래픽을 자랑하는 기계들이 많이 있으나, 이런 그래픽 기능을 지원해 주는 소프트웨어가 부족하기 때문에 소프트웨어가 풍부한 애플이 그래픽에서도 그의 독보적인 위치를 자치할 수 있는 것이다. 게다가 최근에 발표한 애플ⅡGS는 상당히 뛰어난 그래픽을 자랑한다고 하는데, 이것 또한 강력한 소프트웨어로 무장하고 있어 애플의 그래픽 기능은 여러 화제거리를 만들고 있다.

한편 이 더블 고해상도 그래픽은 베이직으로는 사용이 불가능하다. 물론 별도의 유틸리티를 쓰면 가능하기는 하나, 이것은 다음 장에서 설명하기로 하고 이 장에서는 더블 고해상도 그래픽의 하드웨어적 특징과 사용법에 관하여 설명하기로 한다.

1. 왜 더블 고해상도 그래픽이 필요한가

그래픽의 확장은 특히 한글 적용 문제를 안고 있는 우리나라의 컴퓨터 사정에 비추어 볼 때 절실한 것이라고 할 수 있다. 기존 애플Ⅱ+의 280× 192도트의 화면은 영문자를 만들어내는 데에는 큰 문제가 없었으나, 한글의 경우에는 가로 20자, 세로 12줄이라는 아주 적은 양밖에 만들 수 없기 때문에 한글을 사용하는 프로그램의 개발을 더디게 한 요인이기도 하였다. 그런데 가로 해상도가 두 배인 애플Ⅱe 계량형에서는 한글이 가로 40자까지 가능하기 때문에 훨씬 많은 문자를 사용할 수 있게 되었다. 이 점은 애플의 더블 고해상도 그래픽용 한글을 사용해 본 독자라면 잘 알 수 있을 것이다. 애플의 화면을 잘 보면 다음과 같은 특성을 발견할 수 있다. 다음을 실행해보자.

```
10 HGR : HCOLOR = 3
20 HPLOT 10, 10 : HPLOT 11, 10
30 HPLOT 10, 11 : HPLOT 11, 11
```

프로그램에 의하면 애플의 고해상도 화면에서 네 개의 점을 가로 두 개 세로 두 개 연결해 놓은 것이다. 그런데 화면을 자세히 보라. 가로의 두 점은 서로 붙어 있는데, 세로의 두 점은 점과 점 사이에 약간의 간격이 있을 것이다. 이는 애플의 그래픽 특성이 이렇기 때문에 어쩔 수 없는 것이다. 그래서 40컬럼 한글의 글씨를 보면 글자 모양이 좋지 않다.

2. 더블 고해상도 그래픽 모드(DHGR mode)

(1) 더블 고해상도 그래픽 모드의 개관

더블 고해상도 그래픽에는 더블 GR과 더블 HGR이 있는데 여기서는 더블 HGR에 한해 설명하기로 하자.

더블 고해상도 그래픽은 가로 560, 세로 192도트의 해상도를 가진다. 게다가 색번짐이 없는 16색을 쓸 수 있다. 물론 이 색은 네 개의 도트가 한 개의 색을 만들어내므로 16색을 모두 쓰면 가로 해상도는 1/4로 줄어든다. 이에 대해서는 차차 설명하기로 하고 먼저 그림 5. 2를 보자. 이것은 애플의 더블 고해상도 화면의 전체적 레이아웃이 된다.

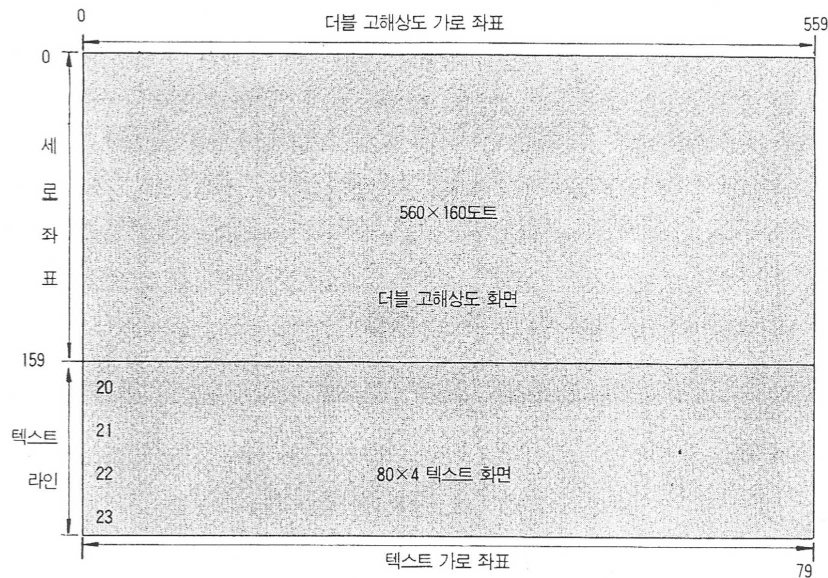


그림 5. 2 더블 고해상도 그래픽의 좌표계

이 그림에서는 아래에 네 줄의 텍스트를 사용하기 때문에 세로 해상도는 192도트가 아니라 160도트가 되었다. 물론 192도트 그래픽 모드로도 사용할 수 있다. 그것은 앞의 128K램 사용과 마찬가지로 소프트 스위치를 써서 정하게 된다.

(2) 화면과 메모리와의 관계

이 더블 고해상도 화면은 다음과 같은 원리로 구성되고 있다. 더블 고해상도 화면이 기존 고해상도 화면의 두 배의 수평 해상도를 가진다는 것과 꼭 128K의 애플Ⅱe에서만 실행된다는 점을 가지고 우리는 한번 생각해 볼 수 있다. 다시 말하면 더블 고해상도 화면은 보조 메모리를 사용하고 있다는 것을 말해준다는 것이다. 다음은 더블 고해상도가 사용하고 있는 애플Ⅱe의 메모리 맵이다.

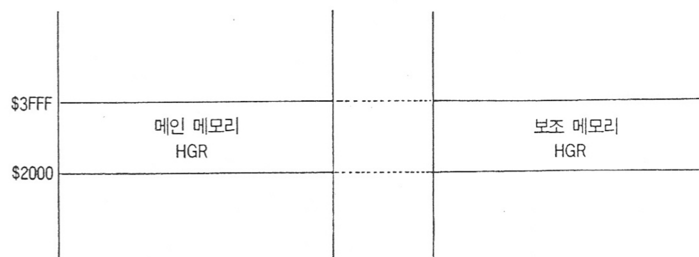


그림 5. 3 애플Ⅱe 더블 고해상도의 메모리 맵

그런데 이 메인 메모리와 보조 메모리 두 개의 영역으로 나뉘어져 있는 더블 고해상도 그래픽은 생각처럼 간단히 되어 있지는 않다. 즉 가로 0-559의 560개 점이 정확히 반(0-279와 280-559)으로 나뉘어져서 애플의 메인/보조 메모리에 대응되어 있으면 참 좋을텐데 사실은 그렇지 않다. 보조 메모리를 시작으로 점이 일곱 개씩 보조/메인 메모리를 번갈아 가며 위치되어 있다는 것이다. 즉 화면 상단의 일곱 개의 점(0-6)은 보조 메모리의 $2000 번지에 대응되고 다음 일곱 개(7-13)는 메인 메모리의 $2000번지에, 그 다음 일곱 개(14-20)는 보조 메모리의 $2001번지에 대응되어 있다는 것이다. 이것을 표로 만든 것이 표 5. 2이다. 아래에 나와있는 숫자(0-6 등)는 바로 그 위에 나와있는 메모리에 대응된다. 0-559까지의 560개의 점 모두를 늘어놓았다.

보조 메모리	메인 메모리	보조 메모리	메인 메모리
0 - 6	7 - 13	280 - 286	287 - 293
14 - 20	21 - 27	294 - 300	301 - 307
28 - 34	35 - 41	308 - 314	315 - 321
42 - 48	49 - 55	322 - 328	329 - 335
56 - 62	63 - 69	336 - 342	343 - 349
70 - 76	77 - 83	350 - 356	357 - 363
84 - 90	91 - 97	364 - 370	371 - 377
98 - 104	105 - 111	378 - 384	385 - 391
112 - 118	119 - 125	392 - 398	399 - 405
126 - 132	133 - 139	406 - 412	413 - 419
140 - 146	147 - 153	420 - 426	427 - 433
154 - 160	161 - 167	434 - 440	441 - 447
168 - 174	175 - 181	448 - 454	455 - 461
182 - 188	189 - 195	462 - 468	469 - 475
196 - 202	203 - 209	476 - 482	483 - 489
210 - 216	217 - 223	490 - 496	497 - 503
224 - 230	231 - 237	504 - 510	511 - 517
238 - 244	245 - 251	518 - 524	525 - 531
252 - 258	259 - 265	532 - 538	539 - 545
266 - 272	273 - 279	546 - 552	553 - 559

표 5. 2 더블 고해상도 그래픽의 가로 560점과 메모리와의 대응

표 5. 2를 보면 더블 고해상도 화면은 사용하기가 상당히 까다롭다는 것을 알 수 있다. 만일에 좌표 0,0에서 559, 191로 직선을 긋는다면 메인 메모리와 보조 메모리를 선택하는 스위치를 80회 정도는 움직여야 하는데, 그렇기 때문에 프로그램의 수행 속도가 상당히 늦게 된다.

그림 5. 4는 더블 고해상도 화면을 만들 때에 화면과 메모리와의 대응을 직접 고해상도 화면 그림으

로 보여주고 있다.



그림 5. 4 메모리와 도트와의 대응

애플의 더블 고해상도 화면이 그림 5. 4와 같기 때문에 이 화면은 가로 14개 도트를 한 묶음으로 보고 이 중 앞 일곱 개를 보조 메모리에, 뒤 일곱 개를 메인 메모리에 넣는다고 생각하면 이해가 좀 쉬울 것이다.

(3) 더블 고해상도 그래픽과 관련된 소프트 스위치

그러면 이번에는 이 복잡한 메모리 구성을 가지고 있는 애플의 더블 고해상도 화면을 관리하는 소프트 스위치를 한번 알아보기로 하자.

① 소프트 스위치

$C000(=49152=−16384)

80STORES－이 번지에 어떤 값을 써 넣으면 80STORE 기능을 세트한다. 이 기능은 뒤에 나오는 PAGE2 소프트 스위치와 관련이 있다.

대응되는 소프트 스위치 : $C000

상태 확인 스위치 : $C018(RD80STORE)

$C001(=49153=−16383)

80STOREC－이 번지에 어떤 값을 써 넣으면 80STORE 기능을 제거한다. 이 기능은 뒤에 나오는 PAGE2 소프트 스위치와 관련이 있다.

대응되는 소프트 스위치 : $C001

상태 확인 스위치 : $C018(RD80STORE)

$C018(=49176=−16360)

RD80STORE－이 번지에 들어 있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트=0 (데이타값<128) : 80STORE 기능이 세트된 상태.
최상위 비트=1 (데이타값>127) : 80STORE 기능이 제거된 상태.

$C00C(=49164=-16372)
80COLC-이 번지에 어떤 값을 써 넣으면 현재의 화면을 40컬럼 화면으로 만든다.
대응되는 소프트 스위치 : $C00D
상태 확인 스위치 : $C01F(RD80COL)

$C00D(=49165=-16371)
80COLS-이 번지에 어떤 값을 써 넣으면 현재의 화면을 80컬럼 화면으로 만든다.
대응되는 소프트 스위치 : $C00C
상태 확인 스위치 : $C01F(RD80COL)

$C01F(=49183=-16353)
RD80COL-이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.
최상위 비트=0 (데이타값<128) : 80COL 기능이 제거된 상태.
최상위 비트=1 (데이타값>127) : 80COL 기능이 세트된 상태.

$C050(=49232=-16304)
TXTC-이 번지에 어떤 값을 써 넣으면 현재의 화면을 그래픽 화면으로 만든다.
대응되는 소프트 스위치 : $C051
상태 확인 스위치 : $C01A(RDTXT)

$C051(=49233=-16303)
TXTS-이 번지에 어떤 값을 써 넣으면 현재의 화면을 텍스트 화면으로 만든다.
대응되는 소프트 스위치 : $C050
상태 확인 스위치 : $C01A(RDTXT)

$C01A(=49178=-16358)
RDTXT-이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.
최상위 비트=0 (데이타값<128) : TXT 기능이 제거된 상태.
최상위 비트=1 (데이타값>127) : TXT 기능이 세트된 상태.

$C052(=49234=-16302)
MIXEDC-이 번지에 어떤 값을 써 넣으면 현재의 화면을 네 줄의 텍스트가 없는 풀 스크린 화면으로 만든다.
대응되는 소프트 스위치 : $C053
상태 확인 스위치 : $C01B(RDMIX)

$C053(=49235=-16301)

MIXEDS – 이 번지에 어떤 값을 써 넣으면 현재의 화면을 네 줄의 텍스트가 있는 혼합 스크린 화면으로 만든다.

대응되는 소프트 스위치 : $C052

상태 확인 스위치 : $C01B (RDMIX)

$C01B(=49179=-16357)

RDMIX – 이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트 = 0 (데이타값<128) : MIXED 기능이 제거된 상태.

최상위 비트 = 1 (데이타값>127) : MIXED 기능이 세트된 상태.

$C054(=49236=-16300)

PAGE2C – 이 소프트 스위치를 읽거나 쓰게 되면 PAGE2 기능을 제거한다. 80STORE가 제거되어 있는 상태에서 이 스위치를 쓰게 되면 그래픽 1/2페이지중 1페이지를 쓰게 되고, 80STORE가 세트되어 있는 상태에서 이 스위치를 사용하면 그래픽 메모리를 메인 메모리의 것으로 사용한다.

대응되는 소프트 스위치 : $C055

상태 확인 스위치 : $C01C(RDPAGE2)

$C055(=49237=-16299)

PAGE2S – 이 소프트 스위치를 읽거나 쓰게 되면 PAGE2 기능을 세트한다. 80STORE가 제거되어 있는 상태에서 이 스위치를 쓰게 되면 그래픽1/2페이지중 2페이지를 쓰게 되고, 80STORE가 세트되어 있는 상태에서 이 스위치를 사용하면 그래픽 메모리를 보조 메모리의 것으로 사용한다.

대응되는 소프트 스위치 : $C054

상태 확인 스위치 : $C01C(RDPAGE2)

$C01C(=49180=-16356)

RDPAGE2 – 이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.

최상위 비트 = 0 (데이타값<128) : PAGE2 기능이 제거된 상태.

최상위 비트 = 1 (데이타값>127) : PAGE2 기능이 세트된 상태.

$C056(=49238=-16298)

HIRESC – 이 번지를 읽거나 쓰면 화면은 저해상도 그래픽 모드로 된다.

대응되는 소프트 스위치 : $C057

상태 확인 스위치 : $C01D(RDHIRES)

$C057(=49239=-16297)

HIRESS – 이 번지를 읽거나 쓰면 화면은 고해상도 그래픽 모드로 된다.

대응되는 소프트 스위치 : $C056
상태 확인 스위치 : $C01D(RDHIRES)

$C01D(=49181=−16355)

RDHIRES−이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.
최상위 비트=0 (데이타값〈128) : HIRES 기능이 제거된 상태.
최상위 비트=1 (데이타값〉127) : HIRES 기능이 세트된 상태.

$C05E(=49246=−16290)

DHIRES 80COLS−이 번지를 읽거나 쓰면 화면은 80컬럼 텍스트와 더블 그래픽 모드로 전환한다.
대응되는 소프트 스위치 : $C05F
상태 확인 스위치 : $C07F(RDDHIRES)

$C05F(=49247=−16289)

DHIRES 80COLC−이 번지를 읽거나 쓰면 화면은 40컬럼 텍스트와 보통 그래픽 모드로 전환한다.
대응되는 소프트 스위치 : $C05E
상태 확인 스위치 : $C07F(RDDHIRES)

$C07F(=49279=−16257)

RDDHIRES−이 번지에 들어있는 값의 최상위 비트(MSB)는 다음과 같은 의미를 가진다.
최상위 비트=0 (데이타값〈128) : DHIRES 80COL 기능이 제거된 상태.
최상위 비트=1 (데이타값〉127) : DHIRES 80COL 기능이 세트된 상태.

② 소프트 스위치의 사용예

이제까지는 더블 고해상도에 대한 소프트 스위치에 관해 알아보았다. 그러면 이제는 본격적으로 더블 고해상도 모드로 들어가는 방법에 대해 설명하고자 한다. 방법은 기존의 고해상도 화면으로 들어가는 방법에다가 몇 가지를 추가한 것에 불과하다.

가. TEXT 소프트 스위치($C050)를 사용하여 그래픽 모드로 들어간다.

나. HIRES 스위치($C057)를 사용하여 고해상도 그래픽 모드로 들어간다.

다. MIXED 스위치를 사용해서 네 줄의 텍스트가 있는 혼합 그래픽 모드($C053)로 만든다. 또는 텍스트가 없는 풀 그래픽 모드($C052)로 만들 수도 있다.

이제까지는 보통 고해상도 화면을 만들었다. 더블 고해상도 화면을 만들기 위해서는 다음의 과정을 더하면 된다.

라. 80COL 스위치를 세트한다($C00D). 즉 화면을 80컬럼 모드로 만든다.

마. DHIRES 스위치를 세트한다.($C05E) 그러면 화면은 더블 고해상도 화면이 되어 있다.

바. 80STORE 스위치($C000)를 세트하여 PAGE2($C054, $C055)의 소프트 스위치를 메인/보조

그래픽 메모리 선택 스위치로 바꾼다.

이렇게 세트된 후에는 현재의 화면은 더블 고해상도 화면으로 바뀌었을 것이다. 화면에 그림을 그릴 때에는 앞에서 말한 메인/보조 메모리의 점 관계를 잘 살펴서 해당 메모리의 해당 번지에 어떤 값을 넣으면 된다. $C054번지는 메인 메모리를, $C055번지는 보조 메모리를 그래픽에서 사용하는 경우가 된다.

지금까지 설명한 것을 이용해서 보통의 40컬럼 텍스트 화면을 더블 고해상도 화면으로 바꾸는 프로그램을 만들어보자. 다음 두 개의 프로그램이 바로 그것인데, 앞의 것은 베이직으로 된 것이고 뒤의 것은 기계어 프로그램이다.

```
10 POKE 49232,0      : REM 그래픽 선택
20 POKE 49235,0      : REM 혼합 그래픽 모드
30 POKE 49239,0      : REM 고해상도 그래픽 모드로
40 POKE 49165,0      : REM 80컬럼 모드로
50 POKE 49152,0      : REM 더블 고해상도 모드로
60 POKE 49152,0      : REM 80STORE를 세트함
```

```
300- 8D 50 C0      STA $C050   ; 그래픽 선택
303- 8D 52 C0      STA $C052   ; 풀 그래픽 모드
306- 8D 57 C0      STA $C057   ; 고해상도 그래픽 모드
309- 8D 0D C0      STA $C00D   ; 80컬럼 그래픽 모드
30C- 8D 5E C0      STA $C05E   ; 더블 고해상도 선택
30F- 8D 00 C0      STA $C000   ; 80STORE 세트
```

그러면 이번에는 더블 고해상도 화면을 한번 지워보자. 다음의 베이직 프로그램이 바로 해주게 된다. 본 프로그램에 앞서 먼저 여기서 사용하는 모니터 롬 루틴과 제로페이지에 대하여 알아보자.

HCLR-이 모니터 루틴은 현재의 애플 고해상도 화면을 지우게 된다. HGR이나 HGR2명령과 관계가 있다.

16진 엔트리 포인트 : $F3F2 (JMP $F3F2)

10진 엔트리 포인트 : 62450 (CALL 62540)

먼저 세트해 줄 곳 : $E6(230)번지 = $20(32) : 고해상 1페이지(HGR)

= $40(64) : 고해상 2페이지(HGR 2)

주의 A와 Y레지스터값은 바뀐다.

이 프로그램의 수행 원리는 다음과 같다. 앞의 것과 같이 더블 그래픽 상태를 만든다. 다음에는 $C055 번지를 건드려 보조 메모리 상태로 놓은 다음 화면을 지우는 서브루틴(HCLR)을 CALL한다. 다음에는 메인 메모리 상태로 놓은 다음에 이 서브루틴으로 한번 더 CALL한다. 자세한 것은 프로그램을 보면서 이해하기 바란다.

```
10 POKE 49232,0    : REM 그래픽 모드 설정.
20 POKE 49234,0    : REM 풀 스크린 모드로 만듬.
30 POKE 49239,0    : REM 고해상도 그래픽 모드로.
40 POKE 49165,0    : REM 80컬럼 모드로 만듬.
50 POKE 49246,0    : REM 더블 고해상도 모드로 만듬.
60 POKE 49152,0    : REM 80STORE를 세트함.
70 POKE 49237,0    : REM 그래픽 메모리는 보조 메모리로.
80 POKE 230,20     : CALL 62450 : REM 보조 메모리 지움.
90 POKE 49236,0    : REM 그래픽 메모리는 메인 메모리로.
100 POKE 230,20    : CALL 62450 : REM 메인 메모리 지움.
```

이렇게 해두면 더블 고해상도 그래픽 영역이 말끔히 지워졌을 것이다.

다음에는 고해상도 화면을 지운 뒤 텍스트 화면으로 돌아오도록 해보자. 문제는 간단하다. 위의 프로그램에 단 두 줄을 추가시키면 되는데, 그에 앞서 먼저 새로운 두 개의 모니터 서브루틴에 대해서 알아보기로 하자.

$FB2F(64303) : SETTXT 서브루틴. 현재의 화면을 텍스트 화면으로 돌린다.

$FC58(64600) : HOME 서브루틴. 텍스트 화면을 지운다.

그러면 아래의 프로그램을 보자. 10-100번까지는 앞의 프로그램과 동일한 것으로서 더블 고해상도 화면을 세트한 다음 그 화면을 지운다. 다음의 110번 문장에서는 텍스트 모드로 돌아가고 120번 행에서는 그 텍스트 화면을 지운다. 110번 문장은 TEXT문으로, 120번 문장은 HOME문으로 대체할 수 있다.

```
10 POKE 49232,0    : REM 그래픽 모드 설정.
20 POKE 49234,0    : REM 풀 스크린 모드로 만듬.
30 POKE 49239,0    : REM 고해상도 그래픽 모드로.
40 POKE 49165,0    : REM 80컬럼 모드로 만듬.
50 POKE 49246,0    : REM 더블 고해상도 모드로 만듬.
```

```
60 POKE 49152,0     : REM 80STORE를 세트함.
70 POKE 49237,0     : REM 그래픽 메모리는 보조 메모리로.
80 POKE 230,20      : CALL 62450 : REM 보조 메모리  지움.
90 POKE 49236,0     : REM 그래픽 메모리는 보조 메모리로.
100 POKE 230,20     : CALL 62450 : REM 메인 메모리 지움.
110 CALL 64303      : REM 텍스트 화면으로 돌린다.
120 CALL 64600      : REM 화면을 지우고 커서를 최상단으로 위치시킨다.
```

프로그램을 실행하게 되면 더블 고해상도 그래픽 화면을 보여준 다음 이것을 지우고 다시 텍스트 화면으로 돌아오게 된다. 그런데 이 텍스트 화면은 아직도 80컬럼 화면이다. 이를 40컬럼으로 돌리려면 PRINT CHR$(17)을 수행하면 된다.

(4) 더블 고해상도 화면에 그림이 그려지는 원리

① DHGR의 원리

이번에는 실지로 더블 고해상도 화면에 그림이 그려지는 방법을 알아보기로 하자. 앞에서 더블 고해상도 화면 영역과 메인/보조 메모리의 대응 관계를 설명한 일이 있는데, 이번에는 하나의 바이트가 어떻게 화면상의 점들과 대응되는지를 설명하고자 한다.

그림 5. 5가 그것을 잘 설명해 준다. 1바이트는 여덟 개 비트로 되어있는데, 그 중 최상위 비트(MSB)를 제외하고 나머지 일곱 개 비트가 화면상의 일곱 개의 점을 나타낸다.

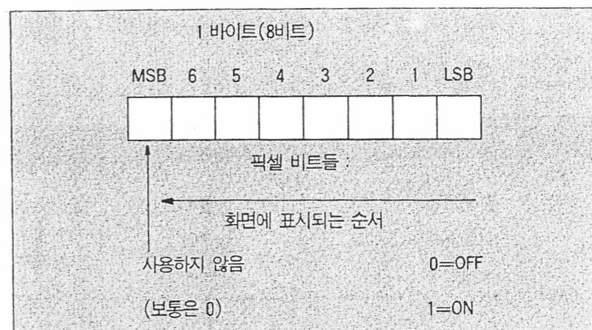


그림 5. 5 1바이트와 화면 7도트와의 관계

보통의 고해상도 화면에는 최상위 비트를 컬러 지정 비트로 사용했었다. 그러나 더블 고해상도 그래픽에서는 이것을 전혀 사용하지 않는다(더블 고해상도에서의 컬러는 다음에 설명하겠다).

보통 8비트는 왼쪽이 최상위 비트(MSB), 오른쪽이 최하위 비트(LSB)이다. 그렇기 때문에 최상위

비트를 제외한 나머지 일곱 비트는 왼쪽의 것부터 차례대로 화면상의 한 점에 대응되어야 하는데 실제로는 그렇지 않고 이들 비트를 전부 바꾸어 놓은 형태로 된다는 것이다. 즉 최하위 비트(LSB)가 화면상의 왼쪽 점을 차지한다는 것이다.

다음 그림은 이것을 알기 쉽게 표현한 것으로 화면상의 "%"문자가 실제로 어떻게 대응되는가를 보여준다.

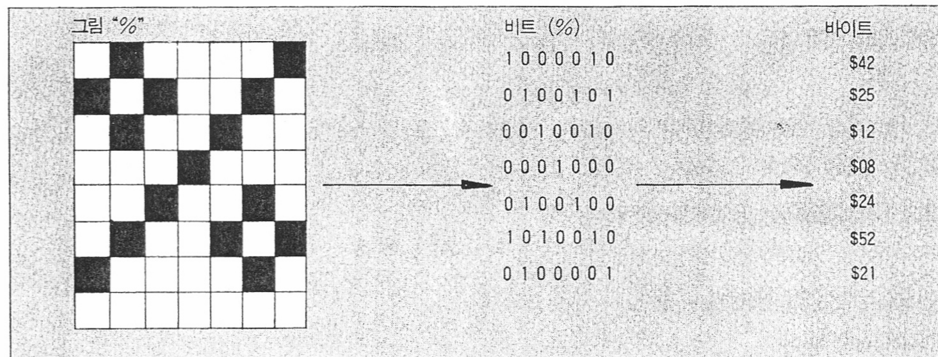


그림 5. 6 화면과 메모리간의 표현 관계

더블 고해상도 화면의 색상 표시는 연속된 네 개의 비트 패턴으로 이루어진다. 모노크롬 모니터에서 보면 이들 네 개의 점은 0일 때 검은색, 1일 때 흰색으로 되어서 560개의 수평 해상도를 가진다. 그런데 컬러 상태일 때에는 이 네 개의 점이 하나의 그룹을 이루므로 수평 해상도는 140점이라고도 말할 수 있으나, 엄밀히 말하면 네 개의 점이 같은 색을 가지게 되는 것이다. 다음 표는 이 비트 패턴과 색상과의 관계를 나타낸 것이다.

비트패턴	색상	비트패턴	색상
0000	Black	1000	Dark blue
0001	Magenta	1001	Purple
0010	Brown	1010	Medium blue
0011	Orange	1011	Light blue
0100	Dark green	1100	Gray 2
0101	Gray 1	1101	Pink
0110	Light green	1110	Aqua
0111	Yellow	1111	White

표 5. 3 비트 패턴과 색상과의 관계

그런데 화면의 일곱 개의 점이 한 바이트를 이루고 네 개의 점이 하나의 색상을 이루기 때문에 하나의 컬러 코드가 정해져 있을 때에는 각 바이트에 넣어지는 코드는 달라지게 된다. 이것은 7과 4의 곱인 28개의 점마다 같은 바이트를 가지게 된다. 먼저 28개 비트를 하나의 컬러 코드로 된 것으로 나열한 다음, 그것을 일곱 개씩 끊어 좌우를 바꾸고 최상위 비트를 0으로 놓아 4바이트로 만든 것이다. 잘 보면 애플의 더블 고해상도 화면의 색상 구성을 알 수 있을 것이다.

② 컬러 코드 요약

다음은 더블 고해상도 컬러 코드를 요약해 놓은 것이다.

색 상	화면(왼쪽→오른쪽)상의 비트 패턴	데이타바이트 형태		
검은색		2 진	16 진	10 진
(black)	0000000000000000000000000000	0000 0000	$00	0
		0000 0000	$00	0
		0000 0000	$00	0
		0000 0000	$00	0
마젠터		2 진	16 진	10 진
(magenta)	0001000100010001000100010001	0000 1000	$08	8
		0001 0001	$11	17
		0010 0010	$22	34
		0100 0100	$44	68
갈색		2 진	16 진	10 진
(brown)	0010001000100010001000100010	0100 0100	$44	68
		0000 1000	$08	8
		0001 0001	$11	17
		0010 0010	$22	34
오렌지		2 진	16 진	10 진
(orange)	0011001100110011001100110011	0100 1100	$4C	76
		0001 1001	$19	25
		0011 0011	$33	51
		0110 0110	$66	102
짙은 녹색		2 진	16 진	10 진
(dark green)	0100010001000100010001000100	0010 0010	$22	34
		0100 0100	$44	68
		0000 1000	$08	8
		0001 0001	$11	17
회색 1		2 진	16 진	10 진
(gray 1)	0101010101010101010101010101	0010 1010	$2A	42

색 상	화면(왼쪽→오른쪽)상의 비트 패턴	데이타바이트 형태		
		0101 0101	$55	85
		0010 1010	$2A	42
		0101 0101	$55	85
밝은 녹색		2 진	16 진	10 진
(light green)	0110011001100110011001100110	0110 0110	$66	102
		0100 1100	$4A	74
		0001 1001	$19	25
		0011 0011	$33	56
황색		2 진	16 진	10 진
(yellow)	0111011101110111011101110111	0110 1110	$6E	110
		0101 1101	$5D	93
		0011 1011	$3B	59
		0111 0111	$77	119
어두운 청색		2 진	16 진	10 진
(dark blue)	1000100010001000100010001000	0001 0001	$11	17
		0010 0010	$22	34
		0100 0100	$44	68
		0000 1000	$08	8
자주색		2 진	16 진	10 진
(purple)	1001100110011001100110011001	0001 1001	$19	25
		0011 0011	$33	51
		0110 0110	$66	102
		0100 1100	$4C	76
(보통) 청색		2 진	16 진	10 진
(medium blue)	1100110011001100110011001100	0011 0011	$33	51
		0110 0110	$66	102
		0100 1100	$4C	76
		0001 1001	$19	25
밝은 청색		2 진	16 진	10 진
(light blue)	1101110111011101110111011101	0011 1011	$3B	59
		0111 0111	$77	119
		0110 1110	$6E	110
		0101 1101	$5D	93
회색 2		2 진	16 진	10 진
(gray 2)	1010101010101010101010101010	0101 0101	$55	85
		0010 1010	$2A	42
		0101 0101	$55	85

색　상	화면(왼쪽→오른쪽)상의 비트 패턴	데이타바이트 형태		
		0010 1010	$2A	42
핑크색		2 진	16 진	10 진
(pink)	1011101110111011101110111011	0101 1101	$5D	93
		0011 1001	$3B	59
		0111 0111	$77	119
		0110 1110	$6E	110
물색		2 진	16 진	10 진
(aqua)	1110111011101110111011101110	0111 0111	$77	119
		0110 1110	$6E	110
		0101 1101	$5D	93
		0011 1011	$3B	59
흰색		2 진	16 진	10 진
(white)	1111111111111111111111111111	1111 1111	$7F	127
		1111 1111	$7F	127
		1111 1111	$7F	127
		1111 1111	$7F	127

위에 서술한 컬러비트들은 각각 1바이트씩 메인과 보조 메모리에 넣어져야 한다. 방법은 앞에 설명한 것과 같다. 그림 5. 7은 앞에서 설명한 더블 고해상도 화면의 메모리와 색상과의 관계를 간단히 요약한 것이다.

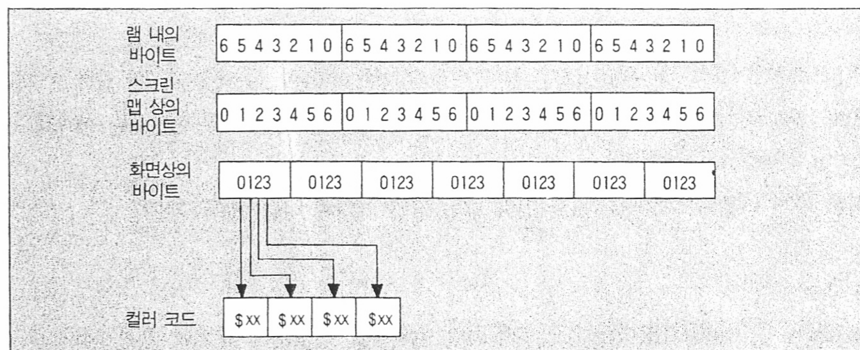


그림 5. 7 메모리와 색상

§ 3. 더블 고해상도 그래픽의 실제 사용

여기에서는 앞에서 설명한 더블 고해상도 화면 원리를 실제로 응용할 수 있는 예를 알아보기로 하자.

1. 애플 베이직과 더블 고해상도 그래픽

안타깝게도 애플 베이직으로는 더블 고해상도 그래픽을 직접 제어할 수 없다. 이것은 애플Ⅱ 시대에 HGR명령어를 내장하지 않은 것과 같은 부류의 것이라고 할 수 있다. 그렇기 때문에 더블 고해상도 그래픽에서 독자들이 할 수 있는 일은 고작 소프트 스위치를 건드려 더블 고해상도 화면을 보는 일에 불과했다.

하지만 더블 고해상도를 베이직에서 아주 쓸 수 없다면 이 책에서 이렇게 별도의 페이지를 써서 나타내지는 않았을 것이다. 사실 애플소프트 베이직에서 더블 고해상도를 쓸 수 없게 한 것은 우리나라 제품에서처럼 메인보드에 내장되어 있는 것이 아니기 때문이다(물론 미국에서도 개량형 애플Ⅱe나 애플Ⅱc에서는 더블 고해상도와 128K가 내장되었지만, Ⅱe의 베이직 인터프리터는 그 이전에 만들어진 것이다).

그렇지만 애플에 없는 모든 기능들을 소프트웨어 업체나 하드웨어 메이커가 만들어 내듯이 이 더블 고해상도를 관리하는 프로그램 또한 유수의 애플 소프트웨어 메이커가 제작했다. 그 중에서 더블 고해상도 그래픽상에서 그림을 그릴 수 있는 대즐 드로우나 비글 그래픽 등이 우리에게 잘 알려져 있는 것이다. 대즐 드로우는 뒤에서 다루기로 하고 여기서는 비글 그래픽을 다루기로 하자.

2. 비글 그래픽을 사용한 더블 고해상도 그래픽

앞에서 이 장의 내용이 베이직상에서 더블 고해상도 명령어를 다루는 것이라고 했다. 그러면 비글 그래픽이란 무엇인가? 왜 이 프로그램과 베이직상의 더블 그래픽이 관계가 있는 것인가?

이유는 다음과 같다. 이 프로그램은 여러가지 기능을 사용할 수 있는데, 그 중에서 더블 저해상도(DGR)와 고해상도(DHGR) 화면에서 마치 베이직의 그래픽 명령어를 사용하는 것처럼 그림을 그릴 수 있는 프로그램이기 때문이다.

자세한 것은 다음에 다루기로 하고 먼저 준비 작업에 대해 서술하기로 하자.

(1) 준비

애플소프트 베이직에서는 저해상도 그래픽과 고해상도 그래픽을 동일한 프로그램에서 사용할 수도 있었다. 예를 들면 다음과 같다.

```
10 GR : HLIN 0,39 AT 10
20 HGR : HPLOT 0,0 TO 279, 159
```

그런데 이 비글 그래픽은 동일한 프로그램에서 사용할 수 없으므로 더블 고해상도를 처리해주는 루틴과 더블 저해상도를 처리해주는 루틴이 나뉘어져 있기 때문에 한 프로그램에서는 한 가지 기능만 쓸 수 있다.

이 더블 저해상도 또는 더블 고해상도를 쓰는 데에는 다음과 같이 한다. 먼저 128K 애플 ⅡE 컴퓨터에 비글 그래픽 디스켓을 넣은 뒤 〈CTRL-OPEN APPLE-RESET〉을 눌러 부팅한다.

그러면 다음 그림과 같은 메뉴 화면이 나오게 된다.

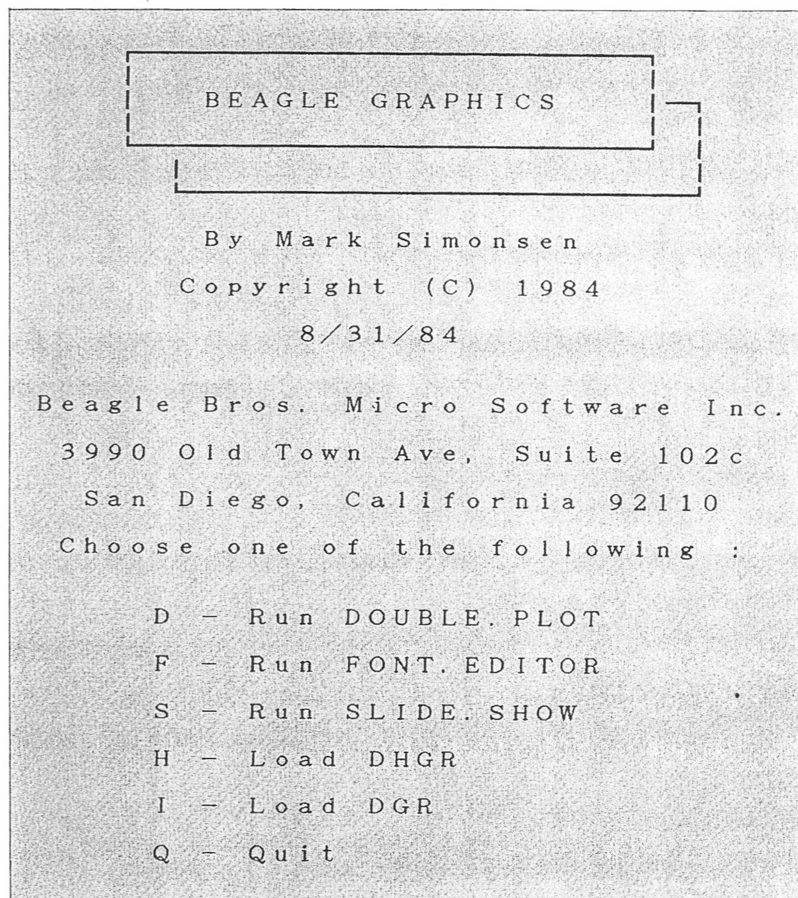


그림 5. 8 메뉴 화면

여기서 L을 누르면 더블 저해상도 모드로 되며, H를 누르면 더블 고해상도 모드로 된다. H 또는 L을 눌러 원하는 기능이 실행된 다음에는 다음과 같은 시작 메시지가 나온다. 그러면 즉시 이 프로그램의 기능을 사용할 수 있다.

사용 방법은 그리 어렵지 않다. 일반 베이직 명령어 앞에 &(Ampersand)를 붙이면 된다. 즉 다음과 같이 한다.

```
10 & HGR
20 & HPLOT 0,0 TO 559, 191
```

여기서 한 가지 알아두어야 할 것이 있는데, 이 더블 고해상도 및 더블 저해상도를 쓸 때에는 현재의 스크린 모드는 언제나 80컬럼 모드로 되어 있어야 한다는 것이다. 그렇기 때문에 이 프로그램을 사용할 때에는 〈ESC－4〉로 40컬럼 화면을 만들지 말라는 것이다. 물론 리셋키를 누르는 것은 두말할 나위도 없다.

그리고 이 프로그램은 ProDOS에서 동작하므로 뒤에 등장할 도스 명령어는 모두 ProDOS라는 것을 알리는 바이다.

그러면 먼저 더블 저해상도부터 사용해 보기로 하자.

(2) 더블 저해상도(double-GR)의 사용

사용법은 앞에서 간단히 설명했으므로 여기서는 명령어와 예제 화면만을 나타내기로 하자. 명령어는 다음과 같다.

GR

형식 : & GR

기능 : 더블 저해상도 화면을 클리어한다. 화면은 80×40도트의 저해상도 화면이 되며 하단에는 네 줄의 텍스트 화면을 사용할 수 있다.

COLOR

형식 : &COLOR =n(n은 색번호 0～15)

기능 : 이것 또한 기존의 저해상도 색상 선택 명령어와 형식이 같다. 16색의 색상도 동일하다.

PLOT

형식 : & PLOT x,y(x,y는 각각 좌표)

기능 : 더블 저해상도 화면에 점을 찍는다. x의 범위는 0－79, y는 0－39이다.

이 세 종류의 명령어를 가지고 다음과 같은 것을 할 수 있다. 다음 프로그램을 보자.

```
10 & GR
20 C=RND(1)*16 : & COLOR = C
30 X=RND(1)*80 : Y=RND(1)*40
```

```
40 & PLOT X, Y : GOTO 20
```

80×40의 더블 저해상도 화면의 임의의 좌표에 점을 찍는 프로그램이다.

VLIN, HLIN

형식 : &VLIN 0, 39 AT 55
& HLIN 0, 79 AT 20

기능 : 베이직에서의 VLIN, HLIN과 기능이 같다. 단 더블 저해상도라는 점이 다를 뿐이다. 여기서 한 가지 생각해 볼 점은 기존의 저해상도 그래픽에서는 가로와 세로의 도트수가 같지만, 더블 저해상도에서는 다르기 때문에 & HLIN, &VLIN 다음의 수들은 잘 생각해야 한다.

다음 프로그램을 실행시켜 보라. 실행 화면은 생략한다.

```
10 & GR
20 FOR A=0 TO 39
30 & COLOR=A
40 & VLIN 0, 39 AT A
50 & VLIN 0, 39 AT 79-A
60 & HLIN 0, 79 AT A
70 & HLIN 0, 79 AT 39-A
80 NEXT A : GOTO 20
```

(3) 더블 고해상도(double-HGR)의 사용

사실 여기서 다루고자 한 것들 중에서 제일 중요한 것들은 더블 저해상도가 아닌 더블 고해상도이다. 더블 저해상도가 기존의 저해상도 명령어를 더블 그래픽용으로 옮긴 것에 불과한 것이라면 이것은 기존의 고해상도 명령어 이외에 몇 개의 명령어를 추가한, 본격적인 것이라 할 수 있다.

HGR

형식 : & HGR

기능 : 더블 고해상도 그래픽 모드로 만든다. 이것은 보통의 고해상도 그래픽에서 HGR명령을 쓰는 것과 같다.

주의 기존의 고해상도 화면에는 HGR과 HGR2의 두 화면이 있지만 더블 고해상도는 하나의 화면밖에 없으므로 & HGR2 같은 명령은 없다.

HCOLOR

형식 : & HCOLOR = n(n은 색번호 0~15)

기능 : 색상을 지정한다. 기존의 것과는 달리 16가지의 색이 있는데, 이것은 앞장의 컬러 코드를 보라.

HPLOT

형식 : & HPLOT x1, y1 TO x2, y2

기능 : 기존의 HPLOT명령과 그 용법 및 기능이 같다. 단 x, y좌표의 범위가 다르다. (x는 0-559, y는 0-191)

XPLOT

형식 : & XPLOT x1, y1 TO x2, y2

기능 : 전혀 보지 못한 명령이기 때문에 좀 자세히 설명하겠다. 간단히 말하면 HPLOT과 XPLOT과의 관계는 DRAW와 XDRAW와의 관계와 같다고 할까? 이 명령의 기능은 화면의 좌표에 그려진 곳을 반대로 그린다는 것이다. 즉 &HPLOT 100, 100으로 명령을 주면 그 점이 찍힌 색의 반대색으로 찍는 것이다. 먼저 그린 점을 지우는 역할을 한다고 할 수 있다.

㊟ 앞으로 설명하는 모든 명령 중 X가 붙은 것은 모두 이렇게 이해하면 된다. 여기서 X는 XOR에서 나온 것이다.

다음 프로그램을 보면 이해가 빠를 것이다.

```
10 & HGR : & HCOLOR = 15
20 & HPLOT 0,0 TO 30,30 : REM 사선을 그음.
30 FOR A=0 TO 30
40 & XPLOT 0,A TO 30, A : REM 주어진 영역 안을 반전시킴.
50 NEXT A
60 END
```

화면상의 특정한 곳에 선을 그은 뒤, 그 선이 위치한 영역을 모두 반전시키게 된다. 자세한 것은 직접 프로그램을 실행시켜 보기 바란다.

DRAW

형식 : & DRAW n AT x, y(n은 도형번호, x, y는 각각의 좌표)

기능 : 도형정의표(shape table)를 참조해서 화면에 그림을 그린다. 기존 애플소프트의 DRAW명령과 그 기능이 같으며, 단지 더블 고해상도에서 쓰인다는 점이 다르다. 물론 전에 HGR에서 만들었던 도형도 그대로 쓸 수 있다.

XDRAW

형식 : & XDRAW n AT x, y(n 은 도형번호, x,.y는 각각의 좌표)

기능 : 도형을 반대색으로 그린다. 그렇기 때문에 DRAW명령으로 그린 도형을 지우는 역할을 한다. 베이직에서의 XDRAW명령과 같은 형식이다.

ROT

형식 : & ROT =n(n은 각도)

기능 : 앞의 DRAW명령으로 그려질 도형의 각도를 정한다. n은 0-255까지의 수이다. 보통은 0이 쓰인다.

SCALE

형식 : & SCALE=n (n은 크기 0-255)

기능 : 앞의 DRAW명령으로 그려질 도형의 크기를 말한다. n이 2, 3, 4가 됨에 따라 도형의 크기도 그 수치만큼 커진다. 단 n이 0이면 도형은 실제의 256배이다. 보통은 1로 둔다.

CLEAR

형식 : & CLEAR

기능 : 애플소프트 베이직에서의 CLEAR명령은 변수를 모두 지우는 역할을 하지만 그것은 더블 고해상도 화면을 지우는 역할을 한다. 지우는 화면의 배경색은 뒤에 나오는 BCOLOR 명령으로 지정해 두어야 한다.

BCOLOR

형식 : & BCOLOR=n (n은 색상 0-15)

기능 : 배경색을 지정한다. 보통은 위의 CLEAR명령과 함께 쓰여지게 된다.

위의 두 개 명령어를 가지고 다음과 같은 예를 실행해 보기 바란다. 컬러 모니터를 가진 독자들은 애플의 더블 고해상도의 화려한 색상에 매료되었을 것이다.

```
10 & HGR
20 FOR A=0 TO 15
30 & BCOLOR=A : & CLEAR
40 NEXT A : GOTO 20
```

BOX

형식 : & BOX(n1, n2)AT x, y

기능 : 기존의 애플 고해상도 명령어에는 없는 것이다. 이 명령의 기능은 화면에 상자를 그리는 기능을 한다. 여기서 n1, n2는 각각 상자의 가로폭, 세로폭이고 x, y는 사각형 좌측 상단의 좌표이다.

XBOX

형식 : & XBOX(n1, n2)AT x,y

기능 : 상자를 그리되 배경색의 반대색으로 그린다. 위의 BOX명령으로 그린 상자를 지우는 기능에 많이 쓰인다.

CIRCLE

형식 : & CIRCLE (n1, n2) AT x, y

기능 : MSX, SPC에서는 많이 보던 명령인데 유독 애플에만 이 명령이 없었다. 이 명령의 정확한 기능은 화면에 원을 그리는 것이다. 여기서 n1, n2는 각각 원의 가로, 세로 반지름이다. 원의 중심은 뒤의 x, y이다(MSX 등과는 형식이 다르다. 이 점 주의하기 바란다). 즉 이 명령은 원 이외에도 타원 등을 그릴 수 있다.

XCIRCLE

형식 : & XCIRCLE (n1, n2) AT x, y

기능 : 위의 원 그리는 명령과 동일한 형식이다. 단지 지정한 색과 반대색으로 그린다는 점이 다르다.

FILL

형식 : & FILL (n1, n2) AT x, y

기능 : MSX컴퓨터의 PAINT명령과 동일한 기능을 한다. 마찬가지로 x, y가 색칠한 곳의 중심 좌표이고 n1, n2는 칠할 색인데, 이 두 개의 색을 혼합하여 쓰게 된다. 이렇기 때문에 256개의 색을 칠할 수 있다.

위의 명령어들을 가지고 재미있는 프로그램을 만들어 보자. 이것은 다른 기종에서 가끔 볼 수 있는 예이다. 한번 실행해 보기 바란다.

```
10 & HGR
20 C=RND(1)*16 : & HCOLOR=C
30 X=RND(1)*560 : Y=RND(1)*192
40 R=RND(1)*40+10
50 & CIRCLE (R, R) AT X, Y
60 & FILL (C, C) AT X, Y
70 GOTO 20
```

TEXT

형식 : & TEXT

기능 : & HGR모드에서 80컬럼 텍스트 모드로 환원한다.

GOTO

형식 : & GOTO x, y

기능 : 그래픽에 웬 GOTO문인가 하고 의아해 한 독자들도 있을 것이다. 이 명령의 기능은 뒤에 소개할 PRINT문으로 화면상에 글씨를 쓸 때, 그 시작점의 좌표를 나타내는 것이다.

PRINT

형식 : & PRINT "문자열"

기능 : 더블 고해상도 화면에 글씨를 쓴다. 단 글씨를 쓰려면 몇 가지 준비 과정이 필요하다. 먼저 원하는 글씨체를 비글 그래픽 디스켓에서 원하는 메모리 번지에 로드시키고 나서 다음을 입력한다.

```
P=PEEK(974)+PEEK(975)*256 : POKE P+3, ML : POKE P+4, MH
```

㈜ 여기서 ML, MH는 로드된 글씨체의 시작 번지의 하위, 상위 바이트이다.

이러한 작업이 끝난 뒤에는 &PRINT명령으로 화면상에 자유자재로 글씨를 쓸 수 있다.

MODE

형식 : & MODE(n)

기능 : & PRINT명령 사용시 글씨의 크기를 정한다. n이 1, 3일 때 보통, 2, 4일 때 확대 문자이다. 보통은 1이다.

NORMAL

형식 : & NORMAL

기능 : & PRINT모드를 해제시킨다(즉 화면에 글씨를 쓸 수 없게 한다).

LOAD

형식 : & LOAD "스트링", "스트링"(단 스트링은 문자 변수로 대체할 수 있다.)

기능 : 더블 고해상도 화면을 디스켓에서 로드한다. 여기서 두 개의 스트링을 필요로 하는 이유는 더블 고해상도 화면은 각각 메인 메모리와 보조 메모리로 나뉘어 있어서 각각을 다른 파일로 만들기 때문이다.

SAVE

형식 : & SAVE "스트링", "스트링"(단 스트링은 문자 변수로 대체할 수 있다.)

기능 : 화면에 있는 더블 고해상도 화면을 디스켓에 세이브한다. 여기서 스트링이 두 개인 이유는 앞의 LOAD의 경우와 같다.

여기까지가 총 22개나 되는 더블 고해상도 명령어이다. 이 명령어군만 보아도 이 애플의 더블 고해

상도 화면이 얼마나 화려하고 강력한가를 알 수 있다. 이제는 "역시 그래픽은 애플"이라는 말을 해도 될 것이다.

그런데 여기서 한 가지 생각해볼 문제가 있다. 더블 그래픽을 쓸 때 언제나 이렇게 비글 그래픽 디스켓을 사용해야 하느냐 하는 것이다. 이 디스켓에서 이 부분만을 뽑아서 따로 쓸 수 없는가 하는 문제를 생각해 보지 않을 수 없다.

여기에 대한 해답은 "OK"이다. 다음과 같은 과정을 거치면 된다.

① 공디스켓 한 장을 준비하여 이것을 ProDOS로 포맷한다. 포맷하는 데에는 ProDOS 마스터 디스켓의 Filer나 Copy Ⅱ Plus 8.2를 쓴다.

② 비글 그래픽 디스켓에서 ProDOS와 BASIC. SYSTEM, DHGR의 세 파일을 복사한다.

③ 다음과 같은 프로그램을 만들어서 STARTUP이라는 파일 이름으로 저장하라.

```
10 D$=CHR$(4):PRINT D$ "BRUN DHGR"
```

이상과 같은 과정을 마치고 이 만들어진 디스켓을 부팅하면 언제든지 더블 고해상도 명령을 쓸 수 있다. 만일에 더블 저해상도도 사용하고 싶으면 같은 방법으로 DGR 파일을 복사해서 만들면 된다. 또는 이 두 개의 파일을 선택하는 메뉴 프로그램을 만들면 된다. ProDOS 상태이므로 여기서 발생하는 문제는 다음의 ProDOS에 관한 장을 읽어보기 바란다.

마지막으로, 베이직에서 만드는 더블 고해상도 화면에 싫증 난 독자라면 대즐 드로우나 애니메이트, 8/6 Paint를 사용하여 본격적인 그림을 그려보기 바란다.

§4. Ⅱe의 대표적인 그래픽 프로그램

이번에는 Ⅱe에서 사용 가능한 그래픽 프로그램들의 사용법을 알아보기로 하자. 여기서는 그 범위를 대즐 드로우(dazzle draw)와 블레이징 패들(blazing paddles)로 한정시키기로 한다. 이밖에도 좋은 프로그램이 많이 있지만 널리 보급되어 구입하기 쉬운 두 가지를 택하여 설명하겠다.

1. 대즐 드로우(dazzle draw)

더블 고해상도에서 사용되는 그래픽 프로그램을 들라면 제1로 꼽히는 것이 바로 이 프로그램이다. 이 프로그램의 장점은 그 이름에서 보듯이 화려한 그래픽과 사용하기에 편리한 점이라고 할 수 있다. 매킨토시에서 많이 사용되는 풀다운 메뉴(pull-down menu)를 사용했다. 단 키보드로는 사용할 수 없다는 단점이 있고 조이스틱 사용시 풀다운 메뉴가 오히려 사용하기 불편한 때도 있다. 그러나 마우스

등 전용 그래픽 입력 도구를 사용하는 경우에는 애플 더블 고해상도 그래픽 기능을 십분 활용할 수 있다. 그러면 먼저 준비 과정을 알아보기로 하자.

(1) 준비

먼저 애플Ⅱe(개량형)나 Ⅱc 컴퓨터가 있어야 하고 하나 이상의 디스크 드라이브와 디스켓이 있어야 한다. 그린 모니터도 좋으나 화려한 더블 고해상도의 그림을 보려면 컬러 모니터가 있어야 할 것이다. 게다가 최소한 조이스틱 이상의 그래픽 입력 도구가 있어야 한다. 참고로 이 프로그램이 사용 가능한 입력 도구는 조이스틱, 마우스, 그래픽 타블렛(graphic tablet), 코알라 패드(koala pad)이다. 그리고 프린터가 있으면 화면을 프린터로 뽑아볼 수 있다. 애플사의 이미지라이터(imagewriter)Ⅱ가 있으면 컬러 프린팅이 가능하다. 물론 우리나라에서 널리 쓰이는 엡슨(epson) 프린터도 지원한다.

이 정도의 준비물이 갖추어졌으면 대즐 드로우 디스켓을 넣고 부팅한다. 처음에 사용하는 입력 도구와 파일의 형태(아무거나 좋다)를 입력한 뒤 메인 메뉴 화면으로 들어가면 된다.

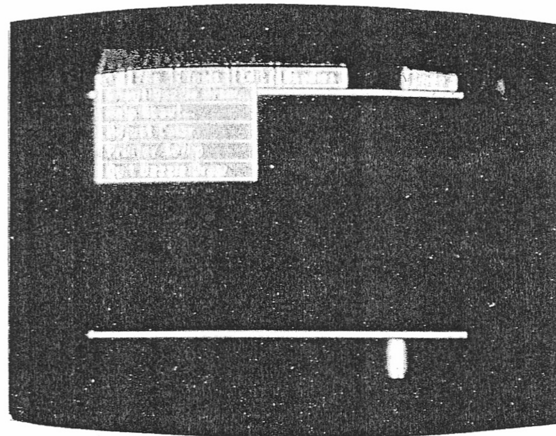


그림 5. 9 메인 메뉴 화면

(2) 사용법

그러면 이 프로그램을 가지고 그림을 그려보자. 먼저 그림 5. 9를 보라. 화면 상단에는 여러가지 메뉴가 있다. 이것을 쓰는 방법은 입력 도구로 원하는 곳에 커서를 위치시킨 후 버튼을 누른다. 그러면 화면 하단에는 그에 해당하는 서브 메뉴가 나오는데, 이 중에서 하나를 고른 다음에 누르고 있던 버튼을 놓으면 된다.

서브 메뉴를 설명하면 다음과 같다.

① BRAND 메뉴

화면 상단의 메뉴판 왼쪽에는 이 프로그램을 만든 브로더번드(Broderbund)사의 로고 표시가 있다(보통 이 부분은 어떤 회사에서 제작하든지 애플 마크가 있는 것이 보통이다. 이 프로그램은 약간 변칙이다). 이 부분에 커서를 위치한 뒤 버튼을 누르면 다음과 같은 서브 메뉴가 나타난다.

- About Dazzle Draw : 제작자의 이름과 제작 일자가 나타난다.
- Help About : 전체 메뉴의 사용 방법이 간단하게 화면에 표시된다.
- Adjust Color : 화면에는 더블 고해상도의 컬러 막대가 나타나는데, 이를 보고 사용자의 컬러 모니터의 색상을 맞춘다.
- Printer Setup : 사용자의 프린터를 설정한다. 프린터의 슬롯 번호와 종류 등을 입력한다.

㊟ 불행하게도 이 프로그램에서만은 프린터 세트를 디스켓에 세이브하지 않는다. 그때그때 지정해 주어야 한다.

- Quit Dazzle Draw : 프로그램의 사용을 끝낼 때 쓴다.

② TOOLS 메뉴

TOOLS 메뉴는 화면에 직접 그림을 그리는 부분이다. 이는 다시 일곱 개의 서브 메뉴로 이루어져 있다. 이를 하나하나씩 설명하면 다음과 같다.

- Paint Brush Tool Window

이것을 고르면 화면의 하단에는 또 다른 메뉴가 나타난다. 여기에는 여러개의 기능이 있는데 각각은 다음과 같다.

–Name : 기능 영역(tool window)의 이름을 나타낸다.

–화면 하단의 색상(또는 패턴) 표시 [이를 Solid Color 또는 Pattern Option이라 부른다] : 현재의 색상(또는 패턴)을 바꾸는 일을 한다.

–Scroll이라고 쓴 막대(스크롤 바) : 화면을 상하로 스크롤시킨다.

–Size : Brush의 모양을 정한다.

–Palette : 16개의 색, 30개의 패턴이 있으며 사용자는 이 중 원하는 것을 고를 수 있다.

–Exit Box : 이 메뉴에서 빠져나와 주 메뉴로 돌아간다.

–Shapes : 여섯 가지 그림 그리는 형태를 정한다.

–Action Box : 지금 정해진 색이 나타나 있다.

- Spray Print(뿌리기 색칠)

모두 네 개의 형태로 되어있는 스프레이 형태를 가지고 화면에 색을 칠한다.

- Flood Fill(메움 색칠)

어느 특정한 색으로 둘러싸인 곳을 지정된 색(또는 패턴)으로 칠해 주는 메뉴이다. 이것을 고르면 화면 하단에는 Fill과 With의 두 가지 기능이 있다는 것을 알 수 있을 것이다. 이 가운데에서 Fill을 선택하면 영역을 정해진 색으로 칠한다. 또한 With는 먼저 정해진 색과 지금 정해져 있는 색을 혼합해서 칠한다.

• Zoom(부분 확대)

작은 지역을 확대해서 세밀한 부분을 확대하는 기능을 한다. 이것은 많은 그래픽 프로그램이 가지고 있지만, 이 프로그램에서는 독특한 기능이 있다. 이는 또한 뒤에 나올 Goodies 메뉴의 격자(grid)를 선택하면 좀더 좋은 작업을 할 수 있다(이 기능은 프로그램 수행시 기본으로 선택되어 있으며 취소할 수도 있다).

• Lines(선 긋기)

화면에 선을 긋는 기능이다. 이것을 선택하면 화면 아래에는 몇 가지 기능을 고를 수 있는 글씨가 나온다.

–Single Lines : 하나의 직선을 긋는다.

–Connection Lines : 연결된 직선을 긋는다.

–Rays : 한 점을 중심으로 퍼져나가는 직선을 그릴 때 사용된다.

• Text(글씨 쓰기)

화면에 글씨를 쓴다. 두 개의 문자체(modern, serif)에 각각 세 개의 문자 형태(plain, bold, italic)가 있고 각각의 문자체에는 두 가지의 글씨 크기(modern은 18과 36, serif는 24와 48)가 있다. 즉 모두 12 종류의 형태가 존재하는 셈이다.

• Shapes(도형)

원, 사각형, 타원, 다각형 등을 그릴 수 있게 해준다.

㈜ 이 Shapes는 애플의 도형정의표(shape table)와는 전혀 다른 것임을 밝혀두는 바이다.

③ EDIT 메뉴

이 EDIT 메뉴 또한 다섯 개의 서브 메뉴로 이루어져 있다. 주요 기능은 화면에 있는 그림을 편집하는 것으로, 하나씩 설명하면 다음과 같다.

• Capture

그림상의 원하는 부분을 Clipboard로 옮겨 다양한 화면을 구성할 수 있다. 또한, 이 부분 화면은 따로 세이브가 가능하다.

이 기능은 또한 세 개의 서브 메뉴가 있는데 각각의 기능은 다음과 같다.

–Invert : 지정된 영역의 색을 반전시킨다.

–Flip Horizontally : 수평으로 그림을 뒤집는다.

–Flip Vertically : 수직으로 그림을 뒤집는다.

• Cut and Paste

화면의 일부를 잘라 다른 곳으로 옮긴다.

• Copy and Paste

화면의 일부를 다른 곳으로 복사한다.

- Clear Section

원하는 부분을 지운다.

- Exchange Colors

원하는 부분의 색을 바꾸어 넣거나 다시 채워넣을 때 쓴다.

④ GOODIES 메뉴

이것은 그림을 그리는 데 필요한 일종의 보조 역할을 해준다고 생각하면 이해가 빠를 것이다. 일곱 개의 서브 메뉴를 알아보자.

- Grid(격자)

앞에서도 잠깐 언급한 바가 있는 이 격자기능은 Zoom 화면의 작업을 돕는 기능을 한다. Zoom 이외에도 패턴의 모양을 바꾸는 Modify Pattern에서도 사용할 수 있다.

- Color(색상 전환)

이 명령은 Zoom이나 Modify Pattern에서 사용한다. 그림의 자세한 작업을 위하여 컬러와 흑백 화면 전환을 한다. 흑백 화면에서는 4도트간 색상 혼합이 없기 때문에 수평 해상도 560도트의 정교한 화면이 가능하며, 이를 가지고 화면 수정을 할 때에는 이전 것보다 세밀한 작동을 할 수 있기 때문이다.

- Mirror(거울)

화면의 한 부분을 거울에 비춘 것처럼 상하 또는 좌우 대칭의 그림을 만들어준다.

- Modify Pattern

앞에서 대즐 드로우에는 모두 30개의 패턴이 있다고 했다. 그런데 이 패턴은 사용자가 마음대로 변경할 수도 있기 때문에 실제로는 그 이상이 있는 것이다. 바로 이 Modify Pattern이 그 일을 해주는 것이다. 마치 Zoom화면과 흡사한 화면으로 패턴을 수정한다.

- View Picture : 메뉴에 가려있는 전체 화면을 보여준다.
- Clear Picture : 현재 화면을 지운다.
- Print Picture : 화면의 그림을 프린터로 인쇄한다. 인쇄를 중단할 때에는 ESC키를 누른다.

  (주) 이전에 프린터를 세트시켜야 한다. 그렇지 않으면 이상한 문자가 출력되는 경우가 있다.

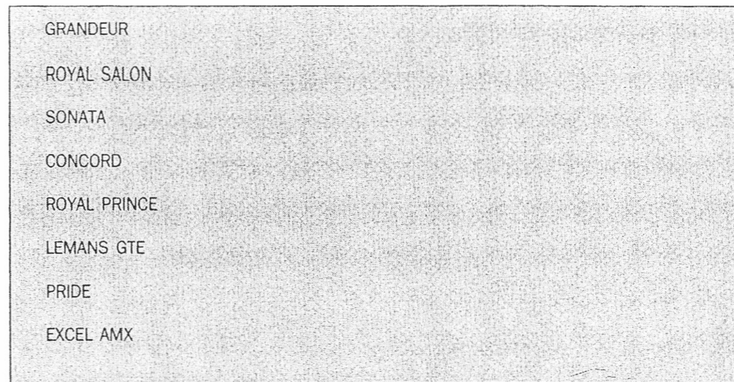
⑤ FILE 메뉴

그림이나 Section(부분 화면)을 데이타 디스켓으로 세이브 또는 로드하는 메뉴이다. 앞에서 대즐 드로우 디스켓을 부팅할 때 두 가지 파일 형태에 대해 잠시 언급했는데, 이것에 대해 설명해보자.

- Easy File : 이 파일 시스템은 보통 도스의 카탈로그처럼 그림 파일을 세이브한다. 따라서 사용하기에는 편하나 한 디스켓에 파일이 많을 때에는 조금 복잡해지는 흠이 있다.
- Professional File : 이 파일은 프로도스 형식의 트리 구조를 만든다. 이것의 기본적인 운영은 프로도스와 거의 흡사하다. 그렇기 때문에 사용하기에는 좀 불편하다.

(주) 위 두 가지 파일 모두 프로도스 파일이다. 단지 Easy는 같은 디렉토리 안에 저장하고 Pro는 다중 디렉토리를 사용한다는 것이다.

이 두 파일의 사용예는 다음과 같다. 먼저 디스켓에 다음과 같은 그림 파일이 있다고 하자. 이 파일의 내용은 이름을 보면 잘 알 수 있을 것이다.


GRANDEUR
ROYAL SALON
SONATA
CONCORD
ROYAL PRINCE
LEMANS GTE
PRIDE
EXCEL AMX

위와 같은 파일이 있을 때 Easy File이라면 이것을 세이브한 순서대로 나타내기 때문에 찾기가 힘들다. 그러나 Professional File이라면 이것들을 비슷한 파일대로 나누어 디렉토리 파일로 만들 수도 있을 것이다.

예를 들면 GRANDEUR, SONATA, EXCEL AMX를 HYUNDAI에, ROYAL SALON, ROYAL PRINCE, LEMANS GTE를 DAEWOO에, CONCORD, PRIDE를 KIA라는 디렉토리 파일에 넣을 수 있는 것이다. 또한 GRANDEUR, ROYAL SALON을 LARGE에, CONCORD, ROYAL PRINCE를 MIDEUM에, LEMANS GTE, PRIDE, EXCEL AMX를 COMPACT에 넣을 수 있다.

이 프로페셔널 파일을 디스켓에서 로드할 때에는 약간 복잡하지만 /디스켓의 볼륨명/디렉토리 파일명/파일명……의 순서대로 입력하면 된다. 이는 일일이 이름을 쳐 넣는 수고를 던다. 이렇게 만든 프로페셔널 파일은 일반 Easy 파일에 비해 로드하기가 힘이 들지만 데이타 디스켓의 정리라는 면에서는 상당히 편리할 것이다. 예를 들면 /KOREAN CAR/KIA/PRIDE와 같이 로드하면 된다.

⑥ MAKE A SLIDE DISK

이것은 여러개의 그림 파일을 묶어서 대즐 드로우의 Demo Slide Show 디스켓처럼 슬라이드 디스켓을 만들어준다.

이상으로 대즐 드로우를 분석해 보았다. 이것은 더블 고해상도를 이용하는 좋은 프로그램의 예가 될 것이다.

2. 블레이징 패들(blazing paddle)

이 블레이징 패들 프로그램은 엄밀히 말해서 애플Ⅱe용은 아니다. 하지만 그 기능의 강력함으로 정평있는 만큼, 이 자리를 빌어 한번 알아보기로 한다.

블레이징 패들은 우리에게 PIXIT나 TAKE-1로 유명한 그래픽프로그램 전문회사인 미국 보드빌(BAUDVILLE)사 작품이다. 그래픽 전문회사인 만큼 그 회사에서 만든 프로그램들은 기능이나 사용의 편리성 등에서 타사의 제품보다 우수하다. 이 프로그램도 예외가 아니어서 다른 그래픽 프로그램에 비해 사용하기가 쉽고 성능이 뛰어나다. 이 프로그램은 국내에서 구하기가 그리 어려운 편은 아닌데, 대부분 파일의 일부가 깨어져 있어 완전한 기능을 발휘하는 것은 구하기 힘들다.

(1) 프로그램의 특징

① 속도가 대단히 빠르다. 원, 사각형, 도형 정의(shape table), 윈도우 등 각각의 기능을 수행할 때, 다른 프로그램에 비해 두 배 이상의 스피드를 낸다. 표 5. 4는 각종 그래픽 프로그램의 처리 속도를 비교한 것인데, 역시 블레이징 패들이 가장 빠른 것을 알 수 있다.

	원 : 반지름 50	원 : 반지름 100	100×100 색칠된 상자	100×100영역 색칠
Blazing Paddles	0.4초	0.6초	0.2초	1.3초
Pixit	11초	22초	–	1초
Koala Painter	1초	1.6초	0.6초	2초
Rainbow Painter	2.5초	2.6초	–	2.4초
The CG System	0.5초	0.9초	–	5초

표 5. 4 그래픽 처리 속도 비교

② 여러가지 입력 장치를 쓸 수 있다. 코알라 패드나 조이스틱 외에 마우스, 그래픽 타블렛, 라이트 펜 등을 사용할 수 있다.

③ 사용하기가 편하다. WINDOW(화면의 일부를 잘라 다른 곳으로 복사하는 것) 기능이 있기 때문에 같은 일을 여러번 반복할 필요가 없다. 또 UNDO 기능(방금 전에 내렸던 명령을 취소하는 것)이 있기 때문에 마음놓고 그림을 수정할 수 있다.

(2) 사용법

먼저 블레이징 패들 디스켓을 넣고 전원을 넣으면 시작 화면이 나오는데, 여기에서 "A"를 누르면 프로그램이 실행된다. 디스켓이 부팅된 후 입력 장치를 선택하라는 말이 나오는데, 〈–, –〉키를 이용하여 가지고 있는 입력 장치를 고른다. 선택이 끝나면 디스켓이 돌고, 곧 메뉴 화면이 나온다.

화면 한가운데에 +표시가 나오는데, 조이스틱(또는 다른 입력 장치)을 움직이면 +표시(이하 커서로 칭함)가 움직인다. 원하는 기능이 있는 위치에 버튼을 놓고 액션 버튼(조이스틱의 경우에는 버튼 0)을 누르면 화면은 그림이 있는 화면으로 바뀌고, 선택한 기능이 실행된다. 그 기능에서 다시 메인 메뉴로 돌아오려면 PAD에서 펜을 떼고 액션 버튼을 2회 누르면 메인 메뉴로 돌아온다. 다른 입력 장치에서는 스페이스바를 누르면 된다. 액션 버튼과 UNDO 버튼의 위치는 헬프 화면을 참조하기 바란다. 각 부분의 기능과 사용법은 다음과 같다.

① CLEAR : 말 그대로 화면을 지운다. 이 기능을 선택하면 화면 가운데에 여덟 개의 색이 나오는데, 원하는 색으로 커서를 옮긴 다음 액션 버튼을 누르면 그 색으로 화면을 지운다. 색이 있는 위치가 아닌 곳에서 버튼을 누르면 화면은 그대로 보존된다.

② WINDOW : 화면의 일부를 잘라서 다른 곳에 복사시킬 때 쓴다. 자른 화면을 디스켓에 세이브시킬 수도 있고 디스켓에서 로드할 수도 있다. 이 기능을 선택하면 화면 가운데에 "CUT", "PASTE"라는 말이 나오는데, CUT은 새로 화면의 일부분을 자른 후 다른 곳에 복사시키는 것이고 PASTE는 이전에 잘라 놓은 부분을 그대로 사용한다. 이것도 역시 그 문자가 없는 곳에서 버튼을 누르면 그 기능은 취소된다.

③ TEXT : 화면의 한 곳에 글씨를 쓴다. 원하는 곳으로 커서를 옮긴 다음 액션 버튼을 누르면 그 곳부터 글씨를 쓸 수 있게 한다. 디스켓에서 원하는 글씨체를 자유롭게 고를 수 있다. 글씨체는 다섯 가지가 준비되어 있다. 또 이 글씨체는 PIXIT나 TAKE-1의 글씨체와 호환성이 있다.

④ SHAPE : 도형 정의된 도형을 화면에 그린다. SHAPE 001이라고 쓴 곳은 여러 개의 도형 중에 원하는 것을 고르는 것으로, 버튼을 누르면 다음 도형으로 넘어간다. ROTATE는 도형의 각도를 정하는데, 버튼을 누르면 각도가 45도씩 증가한다. XDRAW라고 쓰여진 곳은 도형의 DRAW/XDRAW를 결정하는 것이다. DRAW는 화면과 도형을 OR연산을 하여 그리며, XDRAW는 EOR로 그린다. 원하는 색상에 커서를 위치하면 도형의 색을 바꾼다. PICTURE라고 쓴 곳을 선택하면 화면의 원하는 위치에 도형을 그리며, MENU라고 쓴 곳을 선택하거나 UNDO 버튼을 누르면, 이 기능을 취소하고 메뉴로 돌아간다. 한번 설정해 놓은 색상, 각도 등은 계속 유지된다.

⑤ OVAL과 OVAL 2 : OVAL은 화면에 타원을 그린다. 먼저 액션 버튼을 이용해 타원의 중점을 설정한 다음 PAD(또는 다른 입력 장치)로 장축의 길이와 단축의 길이를 정한 후 다시 액션 버튼을 누르면 타원이 그려진다. OVAL은 그냥 타원만을 그리며, OVAL 2는 타원을 그린 후에 내부를 같은 색으로 칠한다. 먼저 말했듯이 이 프로그램의 타원 그리는 속도는 매우 빠르다.

⑥ BOX와 BOX2 : BOX는 화면에 사각형을 그린다. 커서를 움직여 원하는 사각형의 두 대각선 위치를 정하면 사각형이 그려진다. OVAL과 마찬가지로, BOX는 테두리만을 그리고 BOX2는 내부를 칠한다.

⑦ SKETCH : 화면에 자유롭게 그림을 그린다. PAD 위에서 액션 버튼을 누른 채로 펜을 움직이면

그 자리에 점이 찍힌다. 누르고 있으면 계속 점이 찍히기 때문에 펜으로 자유로이 그림을 그릴 수 있다. 이 기능은 조이스틱으로는 쓰기 힘들다.

⑧ DOTS : 화면에 점을 찍는다. 사용 방법은 SKETCH와 같다. 단 DOTS의 경우에는 SKETCH와 달리 버튼 한 번에 한 개의 점만을 찍는다.

⑨ LINE과 LINES : 화면에 선을 긋는다. 먼저 시점을 정하고 종점을 정한다. LINE은 선 하나를 그은 뒤 다시 시점과 종점을 정해야 하며, LINES는 선 하나를 그은 뒤에 종점만을 지정해서 전에 그은 선에 계속 이어 그린다. LINES에서도 스페이스바를 누르면 다시 시점을 잡을 수 있다.

⑩ BRUSHES : 붓의 모양을 바꾼다. 같은 직선을 그어도 붓의 모양에 따라 다른 직선이 그어진다.

⑪ COLOR : 색을 바꾼다. 기본 색깔 여덟 개와 그것을 합성한 약 30가지의 색을 쓸 수 있다.

⑫ FILL : 화면을 색칠한다. 폐곡선으로 막힌 부분을 칠하는데, 경계 색과 칠하는 색이 달라도 색칠이 된다.

⑬ SPRAY : 분무기와 같은 방법으로 화면에 색을 칠한다. 이것은 화면에 특수한 효과를 내기 위해 사용된다.

⑭ ZOOM : 이 기능을 선택하면 현재 커서가 있던 자리에 작은 사각형이 생기고, 그 반대쪽에 그것을 확대한 그림이 표시된다. 이 사각형 안에서 펜을 움직여 원하는 위치에 점을 찍거나 지운다. 즉 점이 찍혀 있는 자리에서 액션 버튼을 누르면 점이 지워지며, 점이 없는 자리에서 버튼을 누르면 점이 찍힌다. 이 기능에는 UNDO 기능을 쓸 수 없으므로 점을 잘못 찍었을 경우에는 같은 방법으로 지워야 한다. 펜을 격자의 끝까지 움직이면 그 부분에 MOVE UP, DOWN 같은 말이 나오는데, 이때 액션 버튼을 누르면 사각형의 격자는 그 방향으로 움직인다.

⑮ PRINTER : 현재 화면에 있는 그림을 프린터로 출력한다. 이 기능은 펜을 움직이지 않고 직접 키보드에서 P를 눌러도 된다. 프린터의 종류를 고른 후 프린트하는데, 프린터가 없을 때에는 이 기능을 쓸 수 없다.

⑯ DISK : 디스켓에서 그림이나 도형 정의, 글씨체 등을 LOAD/SAVE한다. 프린터와 마찬가지로 이 기능도 그냥 D키를 눌러도 된다.

다음 〈−, −〉키를 사용하여 필요한 기능을 선택한 후 리턴키를 누른다. 기능은 다음과 같다.

- SAVE PICTURE : 만들어진 그림을 세이브한다.
- LOAD PICTURE : 디스켓에서 그림을 로드한다. 디스켓에 그림이 없으면 "THERE'S NO PICTURES" 라고 표시된다.
- LOAD SHAPE TABLE : 도형 정의된 도형을 로드한다.
- LOAD CHARACTER SET : 글씨체를 로드한다.
- LOAD WINDOW : WINDOW 기능을 이용해 만든 부분 화면을 디스켓에서 로드한다.
- SAVE WINDOW : WINDOW 기능을 이용해 만든 부분 화면을 디스켓에 세이브한다.

• CHANGE DISK 1 : 현재 사용하고 있는 드라이브 번호를 바꾼다. 이 기능을 선택하면 1이 2로 바뀐다.

• CATALOG DISK : 디스켓의 카탈로그를 본다.

• FORMAT DISK : 디스켓을 초기화(initilize)한다.

• RETURN TO MAIN MENU : 본 메뉴로 돌아간다.

여기서 한 가지 알아둬야 할 것은 보드빌사의 모든 그래픽 프로그램이 그렇듯이 그림을 세이브할 때에는 파일 이름 앞에 "PI."가 붙는다는 사실이다. 따라서 다른 프로그램으로 만든 그림이나 게임 프로그램 등에서 뽑아낸 그림을 이 프로그램에서 사용하려면 "PI."가 붙은 이름(예 : PI. AUTOMAN)으로 바꿔줘야 된다. 마찬가지로 도형 정의 앞에는 "ST."가 , 글씨체에는 "CS."가, 윈도우에는 "WI."가 붙어있으니 착오 없기 바란다. 물론 이 프로그램으로 그림을 세이브할 때는 자동으로 "PI." 등의 문자가 붙는다.

⑰ HELP : 프로그램 사용을 위한 도움말을 제공한다. 각 입력 장치에 관한 설명은 각각 다르나 조이스틱에 관한 설명은 없다. 조이스틱을 쓸 때에는 버튼 0이 액션 버튼이고 버튼 1이 UNDO 버튼이다.

이 프로그램은 앞에서도 잠깐 소개했지만 편리한 UNDO기능이 있다. 어떤 기능을 사용한 후 UNDO 버튼을 누르면 바로 전에 사용한 기능이 취소된다. 이 기능은 매우 편리하지만, 바로 직전에 내린 명령만을 취소한다는 것을 염두에 두기 바란다.

3. 대즐 드로우와 블레이징 패들의 기능 비교

이 두 프로그램은 각기 장단점이 있다. 또한 앞에서 말한 것들 외에도 재미있는 기능이 있다. 표 5. 5는 이 두 프로그램상의 기능을 한눈에 알아볼 수 있게 만든 표이다. 이 표를 보면 이 두 프로그램이 다른 어떤 그래픽 프로그램보다도 기능면과 사용의 편리함에서 뛰어남을 알 수 있다. 단지 단점이라면 키보드로 사용할 수 없다는 점이다.

기능	대즐드로우	블레이징패들	기능	대즐드로우	블레이징패들
그래픽 화면	더블 HGR	HGR	문자 세트의 디스켓 사용	−	Yes
그림의 로드 / 세이브	Yes	Yes	파일의 삭제(delete)	Yes	−
샘플 화면	6개	1개	디스켓의 포맷	Yes	Yes
부분 화면의 로드 / 세이브	Yes	Yes	슬라이드 디스켓 생성	Yes	−
패턴의 로드 / 세이브	Yes	−	카탈로그 디스크	Yes	Yes
도형정의표의 사용	−	Yes	조이스틱 사용 가능	Yes	Yes

기능	대즐드로우	블레이징패들	기능	대즐드로우	블레이징패들
마우스 사용 기능	Yes	Yes	컬러 화면 옵션	Yes	–
키보드 사용 기능	–	–	색상(순수)의 수	16	8
코알라 패드의 사용 기능	Yes	Yes	모니터 색상 조정	Yes	–
그래픽 타블렛의 사용 기능	Yes	Yes	흑백 화면 옵션	Yes	–
라이트펜의 사용 기능	–	Yes	거울 화면(mirror)	Yes	–
펜 브러시의 갯수	24	7	패턴의 변경	Yes	Yes
스프레이 페인트의 기능	Yes	Yes	기준 패턴의 갯수	60	6
색칠 기능(fill)	Yes	Yes	완전한 화면 보기	Yes	Yes
부분 확대(zoom)	Yes	Yes	화면 지움 기능	Yes	Yes
문자체	12	5 이상	지원 가능한 프린터 수	8	11
원, 타원 그림	Yes	Yes	컬러 프린팅 기능	Yes	Yes
연속된 선 긋기	Yes	Yes	도큐멘션 페이지 수	36	29
부분 화면의 복사	Yes	Yes	백업 디스크의 가격	무료	$7.50
색상의 변경	Yes	Yes	디스크 품질 보증 기간	90일	90일
격자(grid)	Yes	–	미국내 판매 가격	$59.95	$49.95

표 5. 5 대즐 드로우와 블레이징 패들의 기능 비교

§5. Ⅱe의 사운드 기능

애플 Ⅱ+에서 애플 Ⅱe로 발전할 때 기능이 많이 향상되었지만 사운드 기능만은 전혀 향상되지 않았다. 그래서 사운드 응용 소프트웨어 중에서 애플Ⅱe 전용 프로그램은 거의 제작되지 않고 있다.

하지만 기존의 소프트웨어로도 충분한 활용을 할 수 있기에 여기에 애플Ⅱ에서의 여러가지 사운드 응용 방법을 소개하고 여러 프로그램과 머킹보드, MIDI에 대해 적어 보겠다. 애플의 사운드 기능이 너무 빈약하기 때문에 사운드에 관심있는 독자는 머킹보드를 꼭 갖추기를 원한다.

1. 음의 발생 원리

애플컴퓨터는 다른 컴퓨터들과는 달리 PSG(Prgrammable Sound Generator)가 내장돼 있지 않아 좀 특이한 방법을 사용한다. 스피커 스위치 번지인 $C030(–16336)에 데이타를 넣거나 빼냄으로써 스피커를 토글시키는 방법이다. $C030번지를 토글시키는 시간을 짧게 할수록 음이 높아지고 길게 할수록 음이 낮아진다.

이 스피커 스위치 번지를 일정한 시간을 두고 계속 토글시키면 어떤 일정한 높이의 음이 연속적으로 발생한다. 그렇기 때문에 볼륨을 조정하려면 하드웨어적인 방법을 사용하거나 소프트웨어적으로 불

편하고 어려운 방법을 사용해야 된다. 이 방법은 뒤에서 자세히 설명하겠다.

그림 5. 10을 보면 음의 발생과 토글과의 연관성을 알 수 있다.

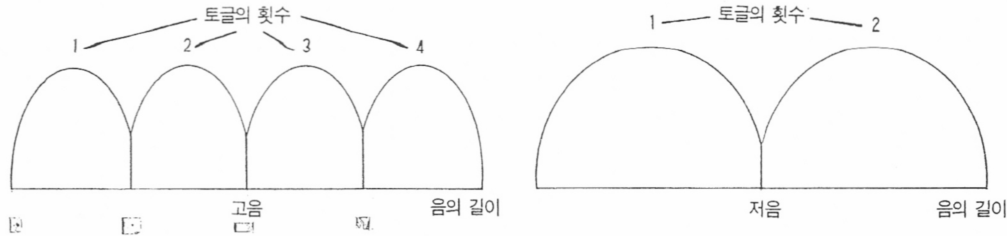


그림 5. 10 음의 발생과 토글과의 관계

원하는 음을 원하는 길이만큼 발생시키려면 그림 5. 10과 같이 임의로 정한 길이 안에 $C030번지에다 토글시키는 횟수를 조정하면 된다. 이렇게 해서 음이 발생하는 과정을 그림 5. 11에 나타내었다.

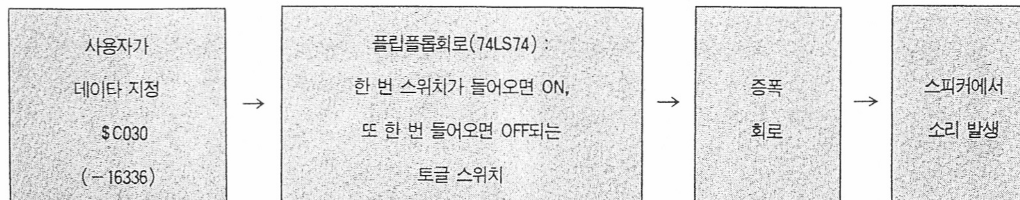


그림 5. 11 음의 발생 과정

(1) 베이직 상태에서의 음의 발생

베이직으로 음을 발생시키면 물론 소리는 나지만 저음밖에 나지 않는다. 베이직이 속도가 느려서 $C030번지를 토글하는 간격이 너무 길기 때문이다. 그러므로 사용자는 자기에게 적합한 기계어 루틴을 제작하여 베이직상에서 호출하는 방법을 사용해야만 소리다운 소리를 들을 수가 있다. 그러면 간단한 뮤직과 사운드를 베이직 상태에서 들어보도록 하자. 하지만 실행은 어디까지나 기계어 상태에서 된다.

리스트 5. 1부터 리스트 5. 6까지는 간단한 음악과 소리를 출력해 주는 프로그램이다. 이 프로그램들은 단순하므로 잘 분석해 보기 바란다.

```
10  REM  ***************************
20  REM    MUSIC EXAMPLE  LIST NO.1
30  REM  ***************************
40  TEXT : HOME
50  PRINT "  YOU ARE MY SUNSHINE. "
60  FOR I = 768 TO 790
70  READ S: POKE I,S
80  NEXT I
90  READ PI: IF PI = 999 THEN  END
100  POKE 0,PI
110  READ LE: POKE 1,LE
120  CALL 768
130 I = I + 1
140  GOTO 90
150  REM  ***************************
160  DATA  224,0,240,3,173,48,192,136
170  DATA  208,4,198,1,240,8,202,208
180  DATA  246,166,0,76,0,3,96
190  REM  ***************************
200  DATA  188,98,140,98,125,98,111,196
210  DATA  111,255,111,98,118,98,111,98
220  DATA  140,196,140,255,140,98,125,98
230  DATA  111,98,104,196,82,196,93,196
240  DATA  104,196,111,196,111,255
250  DATA  140,98,125,98,111,98,104,196
260  DATA  82,255,82,98,93,98,104,98
270  DATA  111,196,140,196,140,196,140,98
280  DATA  125,98,111,255,104,98,125,255
290  DATA  111,98,140,196,140,255
300  DATA  999
```

리스트 5. 1

```
10  REM  ***************************
20  REM    SOUND EXAMPLE  LIST NO.2
30  REM  ***************************
40  FOR I = 770 TO 793
50  READ K: POKE I,K
60  NEXT I
70  TEXT : HOME
80  VTAB 12: HTAB 10
90  PRINT "MACHINE GUN SOUNDS"
100  VTAB 20: HTAB 15
110  PRINT "HIT ANY KEY TO SHOOT !"
120  VTAB 20: HTAB 36
130  GET A$
140  FOR I = 1 TO 7
150 S = 85:L = 30
160  POKE 768,S: POKE 769,L
170  CALL 770
180  NEXT I
190  GOTO 120
200  DATA  202,208,6,174,0,3,173,48
210  DATA  192,136,208,244,206,1,3
220  DATA  240,6,238,0,3,76,2,3,96
```

리스트 5. 2

```
10  REM  ***************************
20  REM    SOUND EXAMPLE  LIST NO.3
30  REM  ***************************
40  FOR I = 771 TO 789
50  READ K: POKE I,K
60  NEXT I
70  TEXT : HOME
80  VTAB 12: HTAB 10
90  PRINT "LASER BEAM SOUNDS"
100  VTAB 20: HTAB 15
110  PRINT "HIT ANY KEY TO SHOOT !"
120  VTAB 20: HTAB 36
130  GET A$
140 LE = 4
150  FOR PI = 254 TO 1 STEP  - 2
160  POKE 0,255 - PI: POKE 1,LE
170  CALL 771
180  NEXT
190  FOR PI = 1 TO 50
200  POKE 0,255 - PI * 5: POKE 1,LE
210  CALL 771
220  NEXT
230  GOTO 120
240  DATA  173,48,192,136,208,4
250  DATA  198,1,240,8,202,208
260  DATA  246,166,0,76,3,3,96
```

리스트 5. 3

```
10  REM   ****************************
20  REM     SOUND EXAMPLE  LIST NO.4
30  REM   ****************************
40  FOR I = 770 TO 793
50  READ K: POKE I,K
60  NEXT I
70  TEXT : HOME
80  VTAB 12: HTAB 10
90  PRINT "BIRD SONG SOUNDS 1"
100  VTAB 20: HTAB 15
110  PRINT "HIT ANY KEY TO SHOOT !"
120  VTAB 20: HTAB 36
130  GET A$
140  FOR I = 1 TO 5
150 S =  RND (1)
160 L = 30
170  POKE 768,S: POKE 769,L
180  CALL 770
190  NEXT I
200  GOTO 120
210  DATA  202,208,6,174,0,3,173,48
220  DATA  192,136,208,244,206,1,3
230  DATA  240,6,238,0,3,76,2,3,96
```

리스트 5. 4

```
10  REM   ****************************
20  REM     SOUND EXAMPLE  LIST NO.5
30  REM   ****************************
40  FOR I = 770 TO 793
50  READ K: POKE I,K
60  NEXT I
70  TEXT : HOME
80  VTAB 12: HTAB 10
90  PRINT "BIRD SONG SOUNDS 2"
100  VTAB 20: HTAB 15
110  PRINT "HIT ANY KEY TO SHOOT !"
120  VTAB 20: HTAB 36
130  GET A$
140  FOR I = 1 TO 5
150 S =  RND (1)
160 L = 30 *  RND (1) * 5
170  POKE 768,S: POKE 769,L
180  CALL 770
190  NEXT I
200  GOTO 120
210  DATA  202,208,6,174,0,3,173,48
220  DATA  192,136,208,244,206,1,3
230  DATA  240,6,238,0,3,76,2,3,96
```

리스트 5. 5

```
10  REM   ****************************
20  REM     SOUND EXAMPLE  LIST NO.6
30  REM   ****************************
40  FOR I = 770 TO 793
50  READ K: POKE I,K
60  NEXT I
70  TEXT : HOME
80  VTAB 12: HTAB 10
90  PRINT "BIRD SONG SOUNDS 3"
100  VTAB 20: HTAB 15
110  PRINT "HIT ANY KEY TO SHOOT !"
120  VTAB 20: HTAB 36
130  GET A$
140  FOR I = 1 TO 6
150 S = 20 *  RND (1) + 1
160 L = 30 *  RND (1) * 5
170  POKE 768,S: POKE 769,L
180  CALL 770
190  NEXT I
200  GOTO 120
210  DATA  202,208,6,174,0,3,173,48
220  DATA  192,136,208,244,206,1,3
230  DATA  240,6,238,0,3,76,2,3,96
```

리스트 5. 6

(2) 기계어 상태에서의 음의 발생

아래의 프로그램을 실행시켜 보라

```
* 300:8D 30 C0    STA $C030
* 303:4C 00 03    JMP $300
* 300G
```

스피커의 토글 간격이 너무 짧아서 스피커가 미처 여기에 대한 반응을 할 여유가 없음을 알 수 있다. 어셈블리어는 너무 속도가 빨라서 소리가 발생되지 않을 정도이다. 따라서 지연 루틴만 만들어 준다면 고음에서 저음까지 자유롭게 다룰 수가 있는 것이다.

우선 리스트 5. 7과 리스트 5. 8을 실행시켜 보라. 리스트 5. 7에서는 고음이, 리스트 5. 8에서는 저음이 나올 것이다. 리스트 5. 7과 리스트 5. 8의 지연루틴을 잘 이용해서 멋진 사운드를 만들어내기 바란다.

```
                ********************
                *  SOUND EXAMPLE   *
                *  LIST NO.7       *
                ********************
                         ORG $300
                *
                *  Y REG - INPUT
                *
0300: A0 20     START    LDY #$20
                *
                *  TWEAKING SPEAKER
                *
0302: 2C 30 C0           BIT $C030
                *
                *  DELAY ROUTINE
                *
0305: 88        LOOP     DEY
0306: D0 FD              BNE LOOP
                *
                *  SOUND AGAIN
                *
0308: 4C 00 03           JMP START
                *
                *  PROGRAM END
                *
                         END
```

리스트 5. 7

```
                ********************
                *  SOUND EXAMPLE   *
                *  LIST NO.8       *
                ********************
                         ORG $300
                *
                *  Y REG - INPUT
                *
0300: A0 C0     START    LDY #$C0
                *
                *  TWEAKING SPEAKER
                *
0302: 2C 30 C0           BIT $C030
                *
                *  DELAY ROUTINE
                *
0305: 88        LOOP     DEY
0306: D0 FD              BNE LOOP
                *
                *  SOUND AGAIN
                *
0308: 4C 00 03           JMP START
                *
                *  PROGRAM END
                *
                         END
```

리스트 5. 8

이번에는 볼륨 조정에 관해 알아보겠다. $C030번지를 토글했을 때 나오는 소리는 일정하게 들리지만 사실은 짧은 시간 동안 음의 크기가 변한다. 음의 크기를 그래프로 보면 그림 5. 12와 같다.

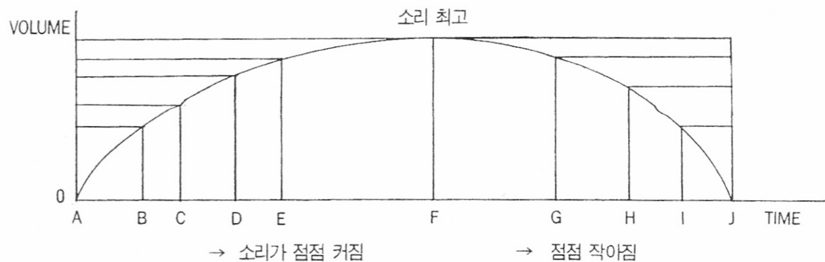


그림 5. 12 볼륨과 토글 시간과의 관계

그림 5. 12에서 알 수 있듯이 A일 때 볼륨이 0이었다가 F일 때 볼륨이 최고가 된다. 그러나 중간에서 끊어주면 볼륨이 줄게 되는 것이다. 그림 5. 13을 보면 지연 루틴을 이용한 볼륨 조정에 관해 알 수 있다.

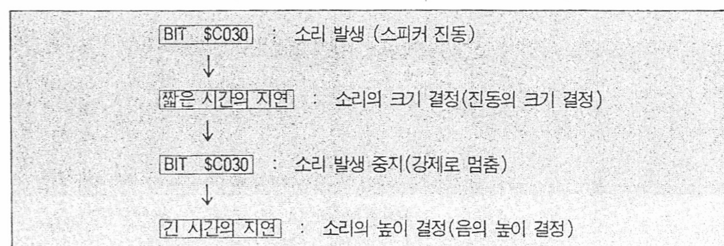


그림 5. 13 볼륨 조정의 원리

리스트 5. 9를 실행시키면 앞에서의 설명을 쉽게 이해할 수 있을 것이다.

2. 애플에 머킹보드를

(1) 머킹보드에 대하여

스위트 마이크로 시스템(Sweet Micro System)사에서 개발한 머킹보드에는 여러가지 종류가 있다. 머킹보드 A와 사운드 I는 상호 호환성이 있고 3화음을 즐길 수 있다. 머킹보드 C와 사운드 Ⅱ는 6화음까지 즐길 수 있으며 음악이나 게임의 효과음에서 자유로이 표현이 가능하고, 또한 소프트웨어적으로 인터럽트의 주기를 쉽게 제어할 수 있으며 많은 테크닉을 구사하여 좋은 음을 표현할 수 있다. 인터럽트의 주기를 조절한다는 의미는 애플에서는 PSG라 불리는 IC를 사용하고 있지 않아서 실제로 음악을 연주할 방법이 없기 때문에 일정한 주기의 주파수를 발생시켜 음을 만드는데, 이 때에 이 주기를

```
              ******************
              *  SOUND EXAMPLE  *
              *  LIST NO.9      *
              ******************
                    ORG $300
              *
              *  VOLUME SETTING
              *
0300: A9 01         LDA #1
0302: 85 FF         STA $FF
              *
              *  LENGTH SETTING
              *
0304: A9 FF         LDA #$FF
0306: 85 FE         STA $FE
              *
              *  TWEAKING SPEAKER
              *
0308: A6 FF   START LDX $FF
030A: A0 FF         LDY #$FF
030C: 2C 30 C0      BIT $C030
              *
              *  MAKE VOLUME
              *
030F: CA      CONT  DEX
0310: D0 FD         BNE CONT
0312: 2C 30 C0      BIT $C030
0315: 88      LOOP  DEY
0316: D0 FD         BNE LOOP
0318: C6 FE         DEC $FE
031A: D0 EC         BNE START
031C: A9 FF         LDA #$FF
031E: 85 FE         STA $FE
              *
              *  VOLUME CONTROL
              *
0320: E6 FF         INC $FF
0322: A5 FF         LDA $FF
0324: C9 0F         CMP #15
0326: D0 E0         BNE START
0328: A9 01         LDA #1
032A: 85 FF         STA $FF
032C: D0 DA         BNE START
              *
              *  PROGRAM END
              *
                    END
```

리스트 5. 9

조절하여 음을 발생시킨다는 의미이다.

원래 머킹보드에는 스피치 IC가 내장되어 있어 SAM 카드를 능가하는 사람의 음성을 출력할 수 있으나 가격이 비싼 관계로 스피치 IC가 내장된 머킹보드는 거의 유통되고 있지 않다. 그리고 프로머킹보드는 앰프를 내장하고 있으며 애플의 스피커 소리까지도 증폭시켜 주므로 모든 음향을 증폭된 스테레오 사운드로 즐길 수 있다. 최근에는 머킹보드와 완전 호환성을 가지며 기능은 훨씬 향상된 파서(P-HASOR) 카드도 나왔다.

(2) MOCKING BOARD DEVELOPER'S MASTER

MOCKING BOARD DEVELOPER'S MASTER에 상당히 많은 기계어 데이타가 들어있다. 직접 실행은 불가능하고 프로그램 상에서만 사용 가능하다. 그러므로 이 데이타를 다른 프로그램에 응용하는 것은 불가능하다.

프로그램 내의 각 메뉴들에 대해 알아보자.

① SPEECH PROGRAM

만일 머킹보드에 스피치 IC가 부착되어 있다면 멋진 음성을 들을 수 있다. 한 가지 아쉬운 점은 SPEECH 신디사이저 프로그램이 비싸므로 수입이 되지 않아 사용자가 자유롭게 스피치 기능을 이용할 수 없다는 것이다.

② SOMTHING SOUND

이 항목을 선택하면 다음과 같은 화면이 몇초 후 출력된다. 이때 사용자는 원하는 효과음을 실행시키면 된다.

SELECTION		SELECTION	
A — GUNSHOOT	: 공기총 소리	I — CLOCK	: 시계의 자명종 소리
B — MACHINE GUN	: 연발 공기총 소리	J — POWER GENERATOR	: 동력기 소리
C — TRAIN	: 기차 엔진 소리	K — LASER	: 레이저 소리
D — HELICOPTER	: 헬리콥터 프로펠러 소리	L — BOMB	: 폭탄 터지는 소리
E — EXPLOSION	: 폭발음	M — MULTIPLE TONES	
F — OCEAN	: 파도 소리, 물결 소리	N — MULTIPLE NOISE Ⅰ	
G — SWISH	: 자동차 엔진 소리	O — MULTIPLE NOISE Ⅱ	
H — PUMP	: 펌프 소리		

③ MUSIC

머킹보드의 음악 기능을 보여주는 예제들이 들어있다. 이 항목을 선택하면 다음과 같은 화면이 나오는데 사용자는 이 중 하나를 선택하면 된다.

A — SUGAR PLUM FAIRY B — MINOTE WALTZ C — POP GOES WEASEL	예제 음악
D — KEY BOARD : 간단한 피아노 연주 기능으로 A부터 K까지의 키를 사용하며 1옥타브밖에 낼 수 없다.	

④ SOUND UTILITY

이 항목은 현재 디스크에 있는 데이타를 로드하여 수정하거나 편집, 추가할 수 있는 음악 작성 에디터이다. P키를 누르면 현재 메모리 안에 있는 음 데이타를 연주한다. B키로 스피커2를 지정할 수 있고 L키로 디스크로부터 데이타를 로드할 수 있다. 이때 C키를 누르면 카탈로그를 볼 수 있다. X키는 메모리 안의 모든 음 데이타를 지워버리고 R키는 스피커 지정을 취소할 수 있게 해준다. S키로 편집

한 음 데이타를 디스크에 세이브할 수 있고 Z키로 이 항목을 빠져나갈 수 있다.

⑤ GRAPHIC SOUND

이 항목은 각종 게임에서의 효과음 출력을 보여주는 예제들을 준비하고 있다. 이 항목을 선택하면 화면이 바뀌면서 다음과 같은 화면이 나온다. ESC키를 누르면 메뉴 화면으로 돌아간다.

A – THE WORM : 화면에 지렁이가 나타나며 꾸물꾸물 기어가는 소리가 난 후 괴성이 나오며 지렁이의 몸이 산산히 부서지고 'OOPS'라는 글자가 나온다.

B – THE LASER : 레이저 음의 특성을 이용해 만든 것으 로 발사음과 폭발음이 준비되어 있다.

C – THE DOOR : 말발굽 소리, 노크 소리, 무시무시한 배경 음악 소리, 문여는 소리 등이 준비되어 있다.

3. 기타 음성 확장 카드들

애플의 음악 기능이 형편없기 때문에 그것을 보충하는 도구들이 많이 제조되었다. 여기서는 그것들에 대해서 서술하고자 한다.

(1) 뮤직 시스템

마운틴 컴퓨터(Mountine Computer)사의 이 제품은 16음성 스테레오 신디사이저 시스템으로, 1790년 이래 컴퓨터를 사용한 최고의 신디사이저로 정상의 자리를 지키고 있다. 각 음성마다 파형과 엔벨로프, 그리고 음량을 프로그램할 수 있으며 8비트 D/A컨버터로 프로그램 작성에 따라 어떠한 곡도 만들 수 있다. 또한 전체적인 음량도 조절할 수 있다. 주파수는 0.5Hz단위로 정의되고 악보의 입력은 HI-RES 화면을 보면서 라이트펜으로 입력시킬 수 있다. 하드웨어는 한 파형 주기를 2백50등분하여 애플 메모리에 입력하며, DMA(Direct Memory/Access)에 따라 이 값을 판독하여 D/A를 통해 출력시킨다. 이 장치는 좌우에 각각 여덟 개의 채널을 가지고 있고 최초 주파수 시간의 50%를 이 소프트웨어가 사용한다. 최고 13KHz까지 출력이 가능한데, 이 시스템으로 음표를 입력하는 것보다 음을 만드는 작업이 더 어려울 것이다. 뮤직 시스템은 9년 전에 만들어진 것인데도 아직 널리 이용되고 있으며 음악적 재능이 없는 사람이라도 쉽게 활용할 수 있다. 사용 전에 스테레오 앰프와 스피커 두 개를 준비해 둘 필요가 있다.

(2) 카나리아 Ⅱ

카나리아Ⅱ는 머킹보드에 비견할 만한 훌륭한 뮤직 카드이다. 9채널을 갖추고 있으며 세 개의 스피커를 사용해 9중화음과 여러가지 음색을 동시에 즐길 수 있다. 마스터 디스켓에 있는 프로그램을 사용하면 기타 반주에 사용되는 여덟 개의 코드와 기타나 피아노에서 쉽게 연주하기 힘든 기교도 간단

히 연주할 수 있다.

또한 스피커로 흘러나오는 음향을 시각적으로도 감상할 수 있으며 음악 연주 도중 연주 속도를 마음대로 지정할 수도 있다. 음악적인 지식이 전혀 없는 사람도 충분히 사용할 수 있으므로 한번 사용해 볼 만한 뮤직 카드이다.

(3) SAM 카드

SAM이란 소프트웨어 오토매틱 마우스(Software Automatic Mouth)의 약자로서 약간의 하드웨어를 사용해 말소리를 합성하는 음성합성기를 뜻한다. SAM 카드는 본체의 슬롯2에 꽂아 사용한다. 음량은 SAM 카드 안에 내장된 조절 나사를 드라이버로 조정한다.

마스터 프로그램에는 SAM, RECITER, SAYIT의 세 종류가 있다. 이 가운데 SAM은 사용자가 원하는 음성을 실제로 발생시켜 주는 프로그램이다. SAM을 이용하면 사용자의 베이직 프로그램에서 각종 메시지를 직접 음성으로 출력시킬 수 있어 더 생동감 있고 친근한 프로그램을 작성할 수 있다. 베이직 프로그램에서 SAM을 이용하는 경우 사용자가 작성하는 프로그램이 SAM 프로그램 영역을 침범하지 않도록 프로그램에서 HIMEM : 29024를 설정해야 한다. 또 음성이 발생되고 있는 도중 리셋키를 눌러서는 안되며 멈추고자 할 때는 CTRL+C키를 눌러준다.

RECITER는 사용자가 입력한 영어 문장을 SAM이 받아들일 수 있는 음소로 변환시키는 프로그램이다. 따라서 직접 소리로 발생시키고자 하는 영어 문장을 입력하면 된다. 이 RECITER 프로그램은 약 4백50개의 규칙을 사용하여 입력한 영어 문장을 SAM프로그램이 인식할 수 있는 음소로 바꾸어 주는데 그 기능이 완벽하지 못하여 입력한 영어 문장과 발생되는 소리에 약간의 차이가 있는 경우도 있다.

SAYIT 프로그램은 소리를 발생시키고자 하는 문장이나 음소를 직접 받아들여 소리로 바꾸어 주는 프로그램이다.

(4) SUPER TALK

음성 합성 장치로 뮤직 시스템을 개발한 마운틴컴퓨터사에서 개발하였다. 마이크에서 입력된 음성을 메모리 안에 계수화(digitize)시켜 저장시키고 출력하는 것 말고도 부속된 오퍼레이팅 시스템으로 자음과 모음을 조합하여 음성을 합성하는 것이 가능하다. 계수화율도 네 종류까지 설정할 수 있는 등 많은 기능을 가지고 있다.

(5) SPEECH LINK

휴리스틱스(HEURISTICS)사에서 개발한 것으로, 64단어까지 인식시킬 수 있는 음성 인식용 보드이다. 프로그램은 단순히 베이직으로도 작성할 수 있고 시스템 소프트를 롬에 내장하고 있기 때문에

프로그래밍이 쉽다는 장점이 있다. 다른 슬롯에 삽입되어 있는 주변장치는 물론 음성으로도 컨트롤할 수 있다.

(6) SHADOW VET

스코트(SCOTT)사에서 개발한 제품으로 SPEECH LINK의 성능을 좀더 높인 것이다. 음성 식별의 정확도가 향상된 동시에 음성 출력을 키 입력과 똑같이 다룰 수 있기 때문에 워드프로세서나 비지칼크 등 기존 소프트웨어의 데이타 입력용으로도 사용할 수 있다.

4. 본격적인 음악에의 이용－MIDI

(1) MIDI 시스템의 구성

MIDI(Musical Instrument Digital Interface)는 한마디로 악기와 악기, 악기와 컴퓨터를 연결하는 것이라고 할 수 있다. 이것은 케이블만 있으면 간단하게 사용할 수 있다. 예를 들어 두 개의 미디 키보드를 케이블로 접속만 하면 한 대의 악기를 연주할 때 다른 한 대의 악기도 자동적으로 연주된다. MIDI는 키보드 조작만으로 음을 다양하게 합성할 수 있어 음악가나 연주가들에게 큰 도움을 준다. 많은 연주가들은 자신의 두 손으로 되도록 많은 악기를 연주하여 훌륭한 연주를 하기를 원한다. MIDI가 이 문제들을 해결하는 역할을 한다. PC내부의 마이크로프로세서와 신디사이저나 드럼 같은 전자 악기를 연결하여 훌륭한 연주를 할 수 있게 된 것이다.

그러나 MIDI를 이용해서 다양한 연주를 할 수 있다는 커다란 매력에도 불구하고 문제점이 없지는 않다. 어떤 음을 낼 수 있는 능력에 한계가 있고 소프트웨어 개발자들이 전자 악기의 무한한 가능성과 능력을 잘 모르고 있다는 점이다. 그러나 MIDI가 컴퓨터에게 목소리를 부여한 것은 커다란 공헌이라 아니할 수 없다.

MIDI는 이전까지 불가능했던 음악 연주의 자동화를 실현함으로써 연주가, 음악 교육자, 음향 기술자들의 작업 형태를 바꾸어 놓았다. 오늘날 모든 신디사이저에는 MIDI 회로가 들어있으며, 많은 업체들이 MIDI 인터페이스와 소프트웨어의 개발에 심혈을 기울이고 있다. 그러면 현재 우리나라 MIDI 사용자들이 가장 많이 사용하고 있는 로랜드(Roland)사의 MPU-401에 대해 알아보겠다.

MPU-401은 MIDI 악기를 운용하는 프로세싱 유니트이다. 전송 방식은 직렬 전송으로 접속 케이블은 5핀 DIN코드를 사용한다. MPU-401은 컴퓨터와 MIDI 악기를 접속하는 장치이다. MPU-401은 컴퓨터와 MIDI 사운드 모듈의 인터프리터 역할을 하는 일종의 마이크로 컴퓨터인데, 컴퓨터로 하여금 음악에 관한 것만 전문적으로 다룰 수 있게도 한다. 이밖에 인터페이스 카드와 소프트웨어가 필요하다. MIDI 시스템의 구성을 그림 5. 14에 나타내었다.

그러면 MIDI 소프트웨어에 대해 알아보기로 하자.

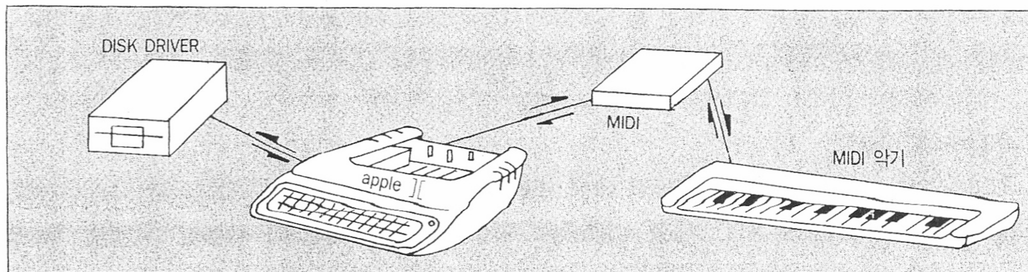


그림 5. 14 애플에서의 MIDI 구성

(2) MIDI 응용 소프트웨어

애플컴퓨터에 사용할 수 있는 MPU-401 용 소프트웨어에는 MRC-APL과 MCP-APL 두 가지가 있다. 자세한 분석은 생략하기로 하겠다.

① MRC-APL

MRC-APL은 애플Ⅱ+ 또는 Ⅱe에 사용할 수 있는 리얼타임 레코더 프로그램이다. 이 프로그램은 여덟 개의 레코딩 트랙을 제공해 주며 Overdubbing, Punch-in, Mixing 같은 편집 기능을 가지고 있어 마치 오케스트라와 같은 사운드를 창조해낼 수 있다.

48KB에서는 약 3천 소절(18,500바이트)을 사용할 수 있으며, 64KB에서는 5천7백 소절(35,000바이트)을 128KB에서는 약 1만6천5백 음표(101,000바이트)를 사용할 수 있다.

② MCP-APL

MCP-APL은 애플Ⅱ+ 또는 Ⅱe에 사용할 수 있는 작곡용 프로그램이다. 이 프로그램은 아주 고도의 연주 기술을 요하는 작곡도 가능하다. 작곡상의 모든 노트를 숫자로 변환하여 애플컴퓨터에 입력시키면 어떠한 연주라도 가능하며, MIDI 악기로 하여금 연주할 수도 있게 한다.

프로그램의 변화나 MIDI 메시지들도 입력될 수 있다. 그리고 복사, 삭제, 삽입 등 몇 개의 편집 기능이 있어 작곡과 배열에 사용할 수 있다.

48KB에서는 2천8백 소절을, 64KB에서는 약 6천9백 소절을, 128KB에서는 약 2만3천3백 소절을 사용할 수 있다.

§6. 애플의 뮤직 소프트웨어

1. MUSIGRAPH

이 프로그램은 애플Ⅱ가 나온 초기에 만들어진 것으로 꽤 오래된 것이지만 쓸만한 프로그램이다.

5옥타브까지 낼 수 있으며 입력과 수정이 편리하다. 하지만 악보의 화면이 적은 점과 단순한 멜로디 입력은 단점으로 지적된다. 그러므로 초보자가 이용하기에 적합한 프로그램이다.

(1) 사용 방법

프로그램을 부팅시키면 이 프로그램에 대한 설명이 나온 다음 디스켓 안의 프로그램을 로드할 것인가를 묻는다. 이때 Y를 누르면 이름을 입력하라는 메시지가 나오고 사용자가 이름을 입력하면 데이타가 로드된 다음 악보가 출력되고 곡이 나온다. Y를 누르지 않고 N을 누르면 본 프로그램이 실행된다.

화면이 바뀌면 TEMPO라는 글자가 깜빡거린다. 이때 3을 누르면 3/4박자를, 6을 누르면 6/8박자를, 4를 누르면 4/4박자를 선택하게 된다. 다음에는 옥타브를 선택하는데 1부터 5 사이이며 1은 저음, 5는 고음이다. NOTE는 음표를 입력하는 것으로 CDEFGAB(도레미파솔라시)를 사용한다. 여기서 ESC키를 누르면 음표가 아니라 쉼표를 선택하게 된다. 다음에는 반음올림과 반음내림을 선택하는데 F는 반음내림, S는 반음올림, N은 정상이다. DURATION은 음표 또는 쉼표의 길이를 지정하는 것으로, 그림 5. 15에 대응키와 박자를 나타내었다.

음표	𝅘𝅥𝅲	𝅘𝅥𝅱	𝅘𝅥𝅯	𝅘𝅥𝅮	𝅘𝅥	𝅗𝅥	𝅝	𝅘𝅥𝅮.	𝅘𝅥.	𝅗𝅥.
박자	$\frac{1}{64}$	$\frac{1}{32}$	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{1}$	점 $\frac{1}{8}$	점 $\frac{1}{4}$	점 $\frac{1}{2}$
대응키	0	1	2	3	4	5	6	7	8	9

그림 5. 15 대응키와 박자

위와 같은 형태로 완전한 음의 구성이 끝나면 해당 음계에 맞는 소리가 나며 악보상에 음이 그려진다. 이렇게 해서 계속 악보를 입력해가면 되는데 만일 입력 도중 수정하려면 ←키를 누르면 처음부터 다시 입력할 수 있다. 악보 입력 도중 지금까지 입력한 음을 들으려면 →키를 누르면 된다. →키를 누르면 악보상에 음이 지나가며 소리가 발생한다. 모든 음이 다 출력되면 다시 듣겠느냐는 메시지가 나온다. 이때 Y를 누르면 다시 들을 수 있고 N을 누르면 프로그램이 종료되며 Y를 누르면 메모리의 내용을 지우지 않겠느냐고 묻는다. 여기서 Y를 누르면 메모리의 내용을 지우지 않고 악보 입력에 들어가며 N을 누르면 새로 악보 입력을 시작할 수 있다.

2. MUSICOMP

MUSICOMP는 MUSIGRAPH의 2탄격인 프로그램으로 MUSIGRAPH의 단점을 보완하면서 사용자의 편리함을 지원하는 프로그램이다. 특히 메뉴중의 EDIT MUSIC은 다른 프로그램에는 없는 독특한 것으로, 입력된 곡을 수정하는 데 아주 효과적으로 쓰일 수 있다.

(1) 사용 방법

프로그램을 부팅시키면 타이틀 화면이 나오며 데모 음악이 출력된다. 음악 출력 도중 ESC 키를 누르거나 음악 출력이 끝나면 메뉴 화면이 나온다. 여기에서는 데모 음악으로 MUSETTE ROMANCE, GAVOTTE, BOURREE, PRELUDE, MINUET, FUGUE 등 여러 종류의 음악이 준비되어 있다. 선택은 ← →를 이용하고 실행은 리턴 키를 누르면 된다. 이 중 ADD NEW MUSIC을 선택하면 본격적인 메뉴 화면으로 들어갈 수 있고 QUIT PROGRAM을 선택하면 프로그램을 빠져나갈 수가 있다.

만일 여기에서 ADD NEW MUSIC을 선택하면 그림 5. 16과 같은 화면이 출력된다.

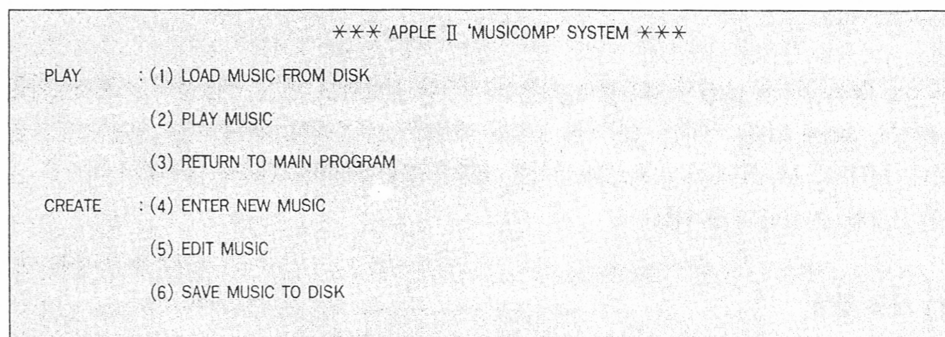


그림 5. 16 MUSICOMP 메뉴 화면

이 메뉴 화면에서 1을 선택한 다음 드라이브를 지정해 주고 파일의 이름을 입력하면 디스켓으로부터 데이타를 로드할 수가 있다. 2를 선택하면 메모리 안의 음악을 출력시켜 준다. 음악 출력 도중 ESC 키를 누른 다음 스페이스바를 누르면 메뉴 화면으로 돌아갈 수 있다.

4를 선택하면 "NEW MUSIC OR CONTINUE?"란 메시지가 나온다. 이때 N을 누르면 메모리 안의 음악을 모두 지우고 새로 시작할 수 있고 C를 누르면 계속 입력할 수 있다. 그러면 또 다른 메시지가 나오는데 여기에서는 Y를 눌러준다.

화면이 바뀌면 악보가 출력되고 화면 아래쪽에 "NOTE #번호?"라는 글자가 나온다. 여기에서 +키(SHIFT+'=')를 누르면 MODE라는 말이 나온다. MODE에서 0을 누르면 NOTES로 들어간다. 1을 누르면 음색을 조정할 수 있고, 2를 누르면 본래의 음색을 가지고 NOTES로 들어간다. NOTES는 음의 길이를 입력하는 것으로 1은 한 박자, 2는 1/2박자, 3은 1/4박자, 4는 1/8박자, 5는 1/16박자이며 앞에 점을 입력하면 점박자가 된다. 만일 MODE에서 1을 입력하면 TIMBRE란 글자가 출력된다. 여기에서 0에서 3까지의 키를 입력함으로써 음색을 조정할 수 있다.

MODE와 NOTES의 입력이 끝나면 "NOTE #번호?"라는 글자가 나오는데 여기에서 해당 음계를 입력시키면 된다. 이렇게 해서 입력이 끝나면 A키를 눌러 NOTE #번호를 증가시켜 나간다. 만일 실

수로 입력했다면 -키를 눌러 수정할 수가 있다. *키를 누르면 음표가 아닌 쉼표를 입력할 수 있다. 입력 도중 ?키를 누르면 메뉴 화면으로 돌아갈 수 있다. 메뉴 화면에서 5를 누르면 입력한 음을 전체적으로 수정할 수가 있다. 5를 누른 다음 시작 라인을 입력하면 거기에서부터 수정을 시작할 수가 있다. EDIT MUSIC에서의 사용키는 E는 수정, I는 삽입, D는 삭제, RETURN키는 다음 라인으로 이동, Q는 메뉴 화면으로 점프, 〉는 EDIT MUSIC의 초기 화면으로의 이동이다.

메뉴 화면에서 6을 선택하면 만든 음악을 디스켓에 세이브할 수 있고 3을 누르면 메인 프로그램으로 다시 갈 수가 있다.

3. MUSIC MAKER

MUSIC MAKER의 소리의 훌륭함은 애플 스피커만 사용하는 다른 어떤 뮤직 소프트웨어들보다 훌륭하다고 알려져 있다. 하지만 에디터의 불편함 때문에 사용자들의 지탄을 받고 있는 프로그램이기도 하다. MUSIC MAKER가 비록 사용하기에 조금은 불편할지라도 음색의 훌륭함이 이러한 단점을 충분히 보상할 수 있다고 생각된다.

(1) 사용 방법

프로그램을 실행시키면 DEMO와 MAKE 중 선택을 요구한다. D를 누르면 데모 화면과 음악이 출력되는데 MUSIC MAKER의 특성을 한눈에 알 수 있다. 여기서 M을 누르면 에디터 화면으로 간다. 그림 5. 17에 MUSIC MAKER의 메뉴 화면을 나타내었다.

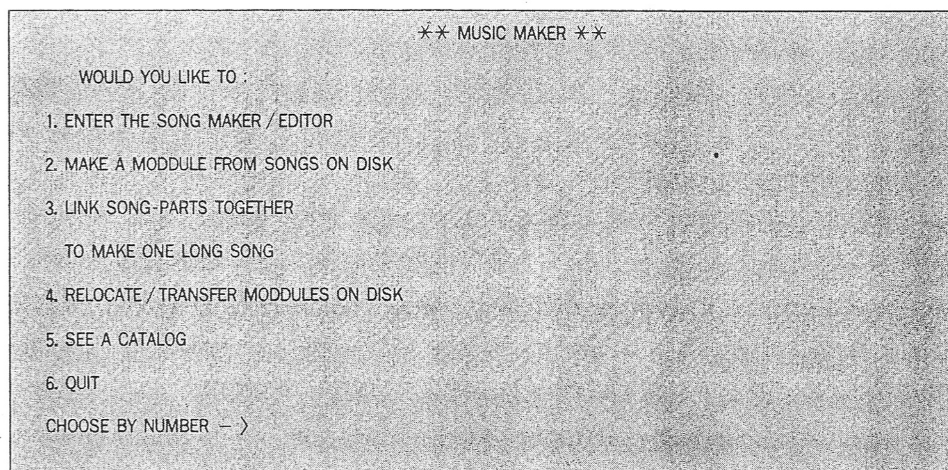


그림 5. 17 MUSIC MAKER의 메뉴 화면

이 메뉴 화면에서 1을 선택하면 음을 만드는 화면으로 넘어간다. SONG MAKER는 설명할 양이 방대하고 HELP 화면이 잘 갖추어져 있으므로 HELP 화면을 참고하기 바란다.

메뉴 화면에서 2를 선택하면 만든 음악을 APPLESOFT, 정수 베이직, 기계어의 세 가지 형태 중 하나로 선택하여 세이브시킬 수 있다. 3은 짧은 음악들을 연결해 긴 곡을 만들 때 효과적으로 쓰일 수 있다. 4는 디스켓 안의 데이타를 재배치하거나 옮길 때 사용한다.

5를 사용하면 디스켓 안의 파일을 전부 보여주고 6은 프로그램을 빠져나간다.

4. ELECTRIC DUET

ELECTRIC DUET은 MUSIC MAKER에 비길 만한 개성있는 소리를 가지고 있다. 큰 스피커를 연결했을 때는 오히려 MUSIC MAKER를 능가한다. 또 입력시 피아노 건반 모양을 보면서 할 수 있어 비교적 쉽고, 음높이나 음길이를 입력한 후에 전체적 혹은 부분적으로 바꿀 수 있어 편리하다. 그러므로 가장 큰 장점은 별다른 노력없이 이중화음을 얻을 수 있다는 것이다.

(1) 사용 방법

프로그램을 실행시키면 메뉴 화면이 다음과 같이 표시된다.

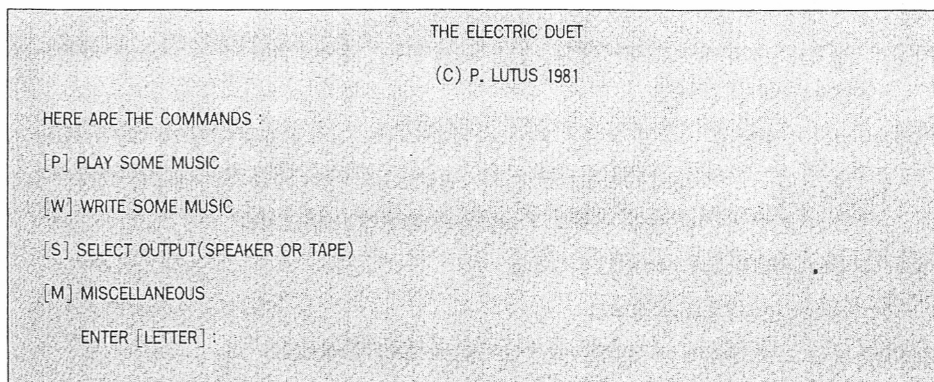


그림 5. 18 ELECTRIC DUET 메뉴 화면

메뉴 화면에서 P를 누르면 다시 피아노와 쥬크박스 중 하나를 선택하게 되는데 피아노를 선택하면 화면에 건반 모양이 나타나 연주를 직접 할 수 있다. 쥬크박스를 선택하면 디스크에 있는 음악들을 표시하고 사용자가 원하는 음악을 연주한다.

메뉴 화면에서 S를 누르면 음악의 출력을 스피커나 카세트 중 한쪽을 선택하게 된다.

① MISCELLANEOUS

이 항목을 선택하면 GENERATE AND SAVE MUSIC PLAYER와 LISTPROGRAM NUMERIC VALUE의 두 기능을 선택할 수 있다. G를 택하고 질문에 대답하면 메인 프로그램 없이도 음악을 듣거나 자신의 플레이어 속에 음악을 넣을 수 있다. 즉 이 기능으로 만들어진 플레이어와 음악 파일을 BLOAD하고 $1E와 $1F에 음악 파일의 시작 번지 하위 바이트, 상위 바이트를 POKE하고 플레이어를 CALL하면 키를 칠 때까지 음악이 연주된다. L을 선택하면 음높이와 음길이에 대한 수치들이 표시되는데 이 값들은 음악을 입력할 때에 도움이 된다.

② WRITE SOME MUSIC

이 항목은 ELECTRIC DUET의 중심 부분으로 음악을 입력하는 기본적인 방법과 도움이 될 만한 사항들을 보여주고 있다. 화면에서 첫번째와 두번째 줄은 명령어와 질문이 표시되는 부분이고 화면 가운데가 입력이 이루어지는 부분이다. 1~64의 수치와 .은 음의 길이를 나타낸다. 아래에 사용키와 대응되는 기능을 나열하였다.

· I.M.J.K : 커서를 상하좌우로 이동시킨다.

· LOAD : 데이타를 메모리로 로드한다. 이때 기존 메모리의 내용이 지워지지 않고 첨가된다. 같은 내용이 반복되는 경우 한 번만 입력해서 세이브시킨 뒤 여러번 로드하면 편리하다

· SAVE : 메모리 안의 내용을 디스크에 기록한다.

· – : 최후에 입력된 줄을 지운다.

· HOP : 입력받은 부분을 처음, 끝, 원하는 번호로 옮긴다.

· PLAY : 메모리 안의 악보를 연주한다. 연주되는 범위는 전체, 현재 위치의 한음, 원하는 번호 등으로 정할 수 있다.

· OPEN : 커서가 위치한 자리에 공간을 만드는 역할을 한다. 이 기능으로 중간 부분의 삽입과 삭제를 할 수 있는데 LOAD와 함께 사용하면 아주 편하다. 중간의 내용을 수정하거나 첨가한 후 다시 OPEN을 선택하면 밀려났던 뒷부분이 합쳐진다.

· ZAP : 메모리의 내용을 모두 지운다.

· X : DOS 커맨드를 사용하게 해준다.

· NOTETRACE : I나 M으로 스크롤할 때 해당되는 음표가 연주된다.

· TRANSPOSE : 입력돼 있는 음들의 높이나 길이를 전체 또는 부분적으로 바꿀 수 있게 해준다.

· * : 메뉴 화면으로 돌아간다.

5. Genius Musician

이 프로그램은 최근 머킹보드와 함께 제공되는 마스터 소프트웨어이다. 10개의 반주 드럼음과 자동 반주 기능 등 다른 프로그램에서는 느낄 수 없는 색다른 기능이 있으나 화음을 지정할 수 없다는 것과

음색을 바꿀 수 없다는 것은 단점으로 지적된다. 하지만 아주 개성있는 소리를 갖고 있으며 사용하기에 편리한 점은 이 프로그램의 큰 장점이다.

(1) 사용 방법

프로그램을 실행시키면 디스크로부터 데이타를 로드하겠느냐는 메시지가 나온다. 디스크 안의 음악을 로드하려면 Y를 누른 다음 파일의 이름을 타이핑하면 된다. N을 누르면 짧은 음악과 함께 아무 키나 누르라는 메시지가 나온다. 이 상태에서 아무 키나 누르면 그림 5. 19와 같은 화면이 출력된다.

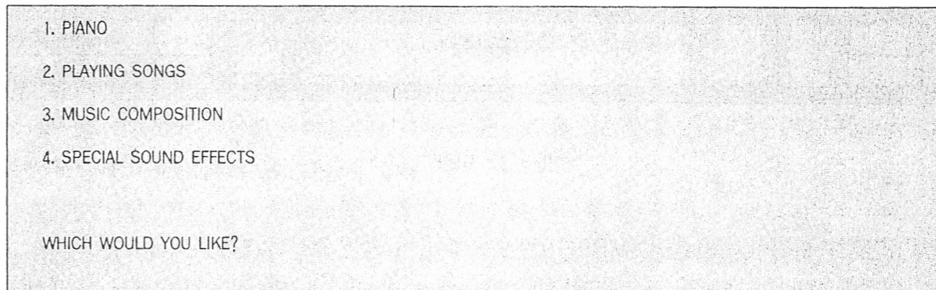


그림 5. 19 Genius Musician의 메뉴 화면

① PIANO

메뉴 화면에서 1을 선택하면 다음과 같은 메시지가 나온다.

HOW MANY BEAT IN BACH MEASURE?
PRESS 2, 3 OR 4, PLEASE!

이것은 박자를 선택하는 것으로 2/4박자곡은 2를, 3/4박자곡은 3을, 4/4박자곡은 4를 누르면 된다. 그림 5. 20에 사용키를 나타내었다.

- ＊(BEAGIN) : 노래를 처음 입력할 때나 입력된 곡을 삭제할 때 사용한다.
- 9(END) : 연주가 끝났거나 입력이 끝났을 때 반드시 이 키를 눌러준다.
- =(REPLAY) : 이 키를 누르면 입력된 곡이 처음부터 연주되며, 곡이 연주되고 있는 동안에 음표가 음(숫자) 위에 표시된다. 이때 →(FASTER)키를 누르면 빠르게, ←(SLOWER)키를 누르면 느리게 곡이 연주된다.
- ?(SHIFT－/) : 한번 누르면 연주가 정지되고 다시 한번 누르면 연주가 계속된다.
- CTRL－I : 잠시 중단하고자 할 때 이 키를 누르고, 중단된 곡을 다시 입력할 때 누르면 된다.
- O(REST) : 쉼표를 의미한다.

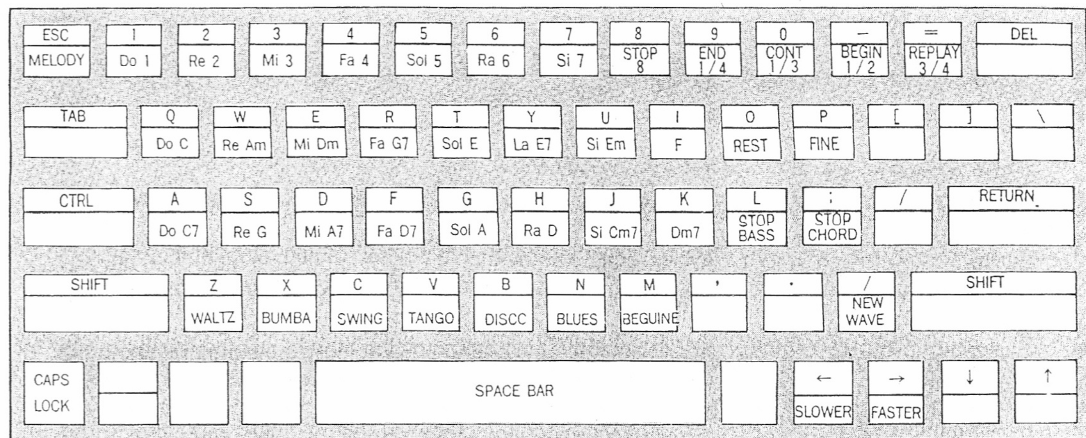


그림 5. 20 사용키

· P (FINE) : 입력이 끝났을 때 MELODY 모드에서 이 키를 누르면 된다.

· L(STOP BASS) : 커서가 RHYTHM에 있을 때, 드럼 반주가 필요없는 마디에서 이 키를 누른다.

· ESC : BEAT 모드에서 ESC키를 누르면 커서는 MELODY로 간다.

· ;(STOP CHORD) : 코드가 필요없을 때 CHORD 모드에서 이 키를 누른다.

입력된 곡을 디스크에 세이브하려면 다음과 같이 한다.

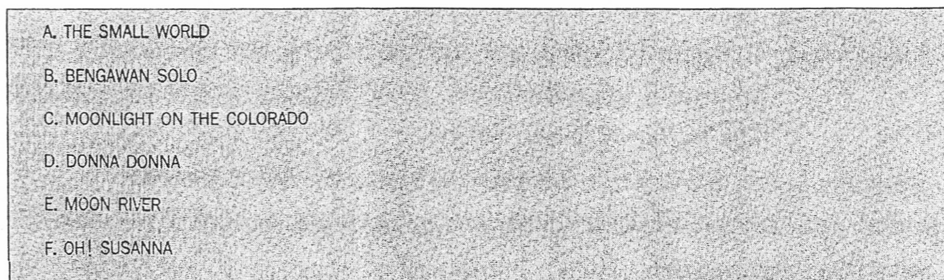
· RESET 또는 CTRL－RESET 키를 누른다.

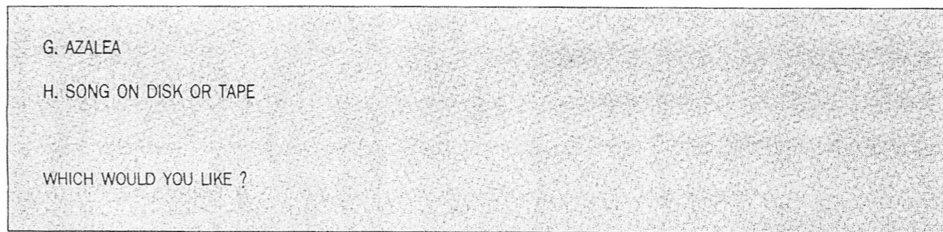
· BSAVE Filename, A$B00, L$1000

길이는 $200에서 $1400까지 곡의 길이에 따라 결정되는데, 일반적으로 $1000이면 충분하다.

② PLAYING SONGS

메뉴 화면에서 2를 누르면 다음과 같은 화면이 출력된다.




G. AZALEA

H. SONG ON DISK OR TAPE

WHICH WOULD YOU LIKE ?

그림 5. 21 PLAYING SONGS의 서브메뉴

이 때 원하는 곡에 해당하는 알파벳을 누르면 연주가 시작된다. 음악을 중단하고 싶을 때는 리턴키를 누르면 된다. H는 디스크에 있는 곡을 로드하여 연주할 때 사용한다.

③ MUSIC COMPOSITION

이 항목을 선택하면 박자를 물어오는데 이때 해당 박자를 입력한다. 사용키는 PIANO와 동일하다. 화면 하단에 나오는 메시지는 다음과 같다.

· RHYTHM : 키보드 하단의 WALTZ, DISCO 등이 이에 해당한다.

· CHORD : 키보드 두번째 줄과 세번째 줄인 G, C, AM 등이 이에 해당한다.

· MELODY : 키보드 위(1~7), 가운데 (Q~U), 아래(A~J)의 3옥타브까지 음을 낼 수 있다.

· BEAT : 첫번째 줄에 해당하고, 음의 길이를 나타낸다.

다음은 입력의 순서에 대해 설명하겠다. 박자를 선택하면 RHYTHM에서 커서가 깜빡거린다. 이 때 드럼 반주를 선택한 다음 스페이스바를 누른다. 그러면 커서는 CHORD에서 깜빡거리는데 여기서 두번째 줄(A~K)과 세번째 줄(Q~U) 중에서 해당하는 CHORD를 선택한다. 다시 스페이스바를 누르면 MELODY에서 커서가 깜빡거린다. MELODY에서 음에 해당하는 키를 누르면 화면 왼쪽 상단부터 차례로 그 음에 해당하는 숫자가 나타나며 숫자의 위와 아래에 옥타브를 나타내는 점이 출력된다. 다시 스페이스바를 누르면 커서는 BEAT에서 깜빡인다. 음의 박자에 해당하는 키를 키보드 상단에서 선택하여 누르면 화면에 자동적으로 그 음의 길이가 출력된다. 다시 스페이스바를 누르면 RHYTHM에서 커서는 깜빡인다. 이와 같이 한 음씩 계속하여 입력하면 된다. 만약 CHORD가 같을 때는 MELODY와 BEAT만을 입력하면 된다. 이때 커서가 BEAT에 ESC 키를 누르면 커서는 MELODY에서 깜빡인다. 이와 같이 계속하면 손쉽게 작곡할 수 있다.

④ SPECIAL SOUND EFFECTS

이 항목을 선택하면 아래와 같은 여덟 가지의 특수 음향 효과를 들을 수 있다.

· GUNSHOT : 총소리

· EXPLOSION : 폭발음

· LASER GUNSHOT : 레이저 총소리

· WHISTLING BOMB : 폭탄 날아가는 소리

· WOLF WHISTLE : 늑대 울음 소리
· SOUND OF WAR : 전장에서의 폭발음
· FIRING ROCKET : 우주선 발사음
· PUCKMAN : 펑크맨 음향

6. MUSIC MASTER

MUSIC MASTER는 원래 카나리아II용 MUSIC 소프트웨어로 개발된 프로그램이다. 이것을 머킹보드로 컨버전하여 사용할 수 있게 만들었다. 아주 우수한 프로그램이므로 사용자가 원하는 만큼 충분한 음향을 제공할 것이다.

(1) 사용 방법

프로그램을 실행시키면 오선지가 화면에 출력되며 악보를 입력할 모든 준비가 갖추어져 입력을 요구한다. 그러면 화면에 나오는 부호에 대해 설명하겠다.

- REST : 쉼표를 오선지에 프린트한다. 이 기호에 화살표를 맞추고 쉼표에 맞는 음표 부호에 화살표를 맞추어 쉼표를 프린트한다.
- 음표 지정 : ↑표시를 원하는 음표 밑에 놓는다. 점음표는 우선 원하는 음표와 점에 각각 화살표를 위치시킨다. 또 셋잇단음표 등은 원하는 각 음표와 3표시된 문자를 같이 지정한 후 오선지상에 기록한다.
- 임시표 지정 : 음표의 길이와 위치를 먼저 정한 후 다음에 임시표를 추가한다.
- 커서키 : ←, →키를 사용해 오선지상의 커서를 앞이나 뒤로 이동시킨다. 데이타의 처음이나 끝일 경우 '삑'소리가 난다.
- INS : 화면의 입력 커서로 입력한다. INS라고 타이프하고 음표 지정을 하면 악보의 커서 위치 다음부터 지정한 음표가 들어가게 되며 이전의 커서 뒤의 데이타는 계속 뒤로 밀리게 된다.
- DEL : 지우고자 하는 음표의 앞에 커서를 위치시킨 후 'DEL : 갯수'로 입력한다.
- 이음줄 : 두 개 음의 음을 부드럽게 이어주는 구실을 한다. 먼저 한 음을 적어준 후 'TIE'를 선택하고 버튼을 누르면 커서가 먼저의 음표의 위치로 이동하고 다시 한 번 누르면 이음줄이 생긴다.
- MEASURE : 악보의 마디 번호를 표시한다. 키보드상에서 'MEASURE : 갯수'하여 에디트하면 훨씬 사용하기가 편리하다.
- PART : 현재 사용하는 화음을 표시한다. 0~5까지 6중화음을 이용한다.

• FREE : 남아있는 메모리 용량을 표시한다.

다음은 악보 입력에 대해 알아보겠다. 메뉴의 선택은 버튼 0을 사용하며, 버튼 1은 악보상의 커서를 위, 아래로 움직이게 한다. 또 지정한 음표, 쉼표 등을 표시할 때 이 버튼을 사용한다. 먼저 NEW를 타이프하면 "NUMBER OF PARTS"라는 메시지가 나오는데 이때 알맞은 화음 수를 입력한다. 다음 SUGGESTED SPEED에서는 이음의 스피드를 1에서 255사이의 숫자로 입력한다.

• KEY : 장조, 단조를 입력하는 것으로 F는 ♭, S는 #이다. 만약 바장조 ♭를 입력하려면 'KEY : 1F'라고 타이프하면 된다.
• TIME : 박자를 선택하는 것으로 미지정시는 4/4가 디폴트이지만, 바꿀 경우 'TIME : 박자'를 타이프한다.
• QUARTER : 메트로놈의 빠르기를 결정하는 것으로 알맞은 값을 'QUARTER : 숫자'로 입력한다. 미지정시는 240이다.
• TRANSPOSE : 음을 전체적으로 높이거나 낮추는 데 사용된다. 이것은 24를 한 옥타브로 사용한다.
• FILE LOAD : 파일을 로드하는 명령으로 'LO : 파일 이름'의 형식을 사용한다.
• FILE SAVE : 파일을 세이브하는 명령으로 'SA : 파일 이름'의 형식을 사용한다.
• PLAY : 음악을 연주할 때 사용한다.
• STEREO : 각 3화음씩 좌우 스피커를 지정할 때 사용한다.
• GOTO : 어떠한 마디를 지정한 후(MEASURE) 다른 PART의 그 마디를 보려면 'GOTO : 번호'라고 타이프한다. 번호는 0에서 8사이의 값이다.
• PART : 지정된 PART의 맨 처음으로 간다.
• SUBROUTINE : 악보상에 도돌이표가 있을 때 사용한다. 지정은 0에서 99까지 가능하며 주의점은 꼭 0, 1, 2 식으로 순서대로 지정되어야 한다. 'PART : 번호'에 알맞는 데이타를 넣고 'SUB : 번호'를 입력하면 새로운 악보가 화면에 나오고 화면 아래에는 'SUB : 번호'로 표시된다.
• ENVELOPES : ATTACK은 한 음이 발생한 후 그 진폭이 최대가 될 때까지의 시간이다. DECAY는 ATTACK에서 지정했던 시간 동안 지속적으로 강세를 나타낸다. SUSTAIN은 단순히 볼륨의 값을 지정하는 것이다.

제 6 장

기타 애플 IIe의 특징

§1. 램디스크
§2. 한글 워드프로세서 "작은별" V 1. 1

§1. 램디스크

1. 개요

(1) 램디스크란

애플Ⅱe 컴퓨터에서보다 진보된 요소중 하나가 바로 이 램디스크일 것이다. 램디스크는 플로피디스크 드라이브보다 빠른 속도로 움직인다. 그래서 플로피디스크 드라이브(이하 디스크 드라이브)의 속도가 느리다고 불평하는 사람이 램디스크를 쓴다면 그 속도에 감탄할 것이다. DOS 3.3에서는 지원되지 않았던 램디스크가 ProDOS에서는 지원이 된다. 그리고 램디스크의 용량은 64K로부터 1MB까지 확장이 가능하다.

램디스크를 사용할 수 있는 유틸리티에는 크게 램디스크 프로그램과 파일을 램디스크에 넣는 것과 램디스크를 이용하여 디스크를 복사하는 것이 있다.

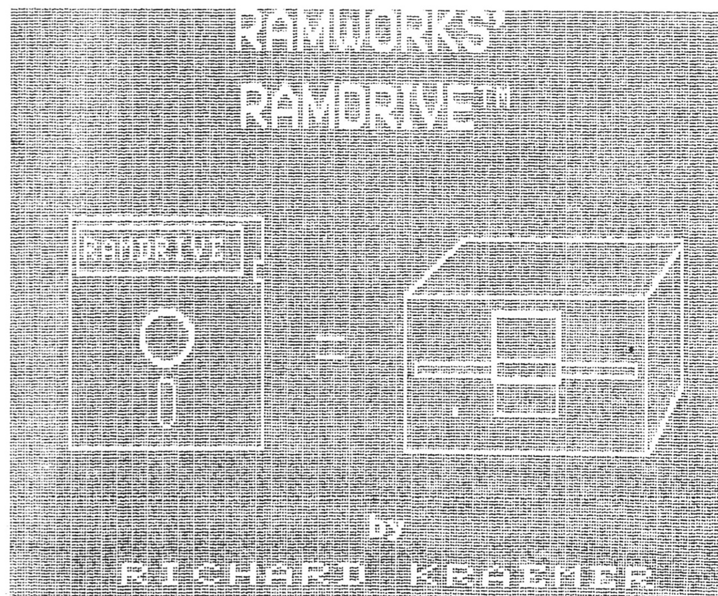


그림 6.1 램드라이브 프로그램의 초기 화면

(2) 램디스크의 용량

Ⅱe의 램디스크는 램워크(RAMwork)가 없는 상태에서의 용량은 디스크의 225섹터 정도이다. 이 정도면 짧은 프로그램 몇 개는 집어넣을 수 있을 것이다.

표6.1은 드라이브와 램디스크의 속도를 비교한 것이다.

종 류	DOS 3.3(변경하지 않은 것)		DOS 3.3(SPEED OS*를 사용한 경우)	
	save	load	save	load
DISKⅡ	40 or 44	32	13 or 20	7
RAMDRIVE	10	10	1.7	0.8

* 위에서의 SPEED OS는 RAMDRIVE 디스켓 내에 있는 것인데 실행은 'BRUN SPEED OS' 하면 된다.

표 6.1 드라이브와 램디스크의 속도 비교

표6.2는 램워크가 있을 경우의 램디스크의 용량이다.

Memory	Free Sectors						Memory	Free Sectros					
Size	D1	D2	D3	D4	D5	D6	Size	D1	D2	D3	D4	D5	D6
64K	234	-	-	-	-	-	576K	744	749	749	-	-	-
128K	489	-	-	-	-	-	640K	744	749	749	239	-	-
192K	744	-	-	-	-	-	704K	744	749	749	494	-	-
256K	744	239	-	-	-	-	768K	744	749	749	749	-	-
320K	744	494	-	-	-	-	832K	744	749	749	749	239	-
384K	744	749	-	-	-	-	896K	744	749	749	749	494	-
448K	744	749	239	-	-	-	960K	744	749	749	749	749	-
512K	744	749	494	-	-	-	1024K	744	749	749	749	749	239

표 6.2 메모리 용량과 디스크 드라이브 용량과의 관계

위의 표에서 보듯이 램용량이 크면 클수록 디스크의 용량도 커짐을 알 수 있다. 램디스크는 DOS 3.3과 같이 최대로 지정할 수 있는 파일의 갯수는 105개이다.

2. 사용법

(1) 부팅

사용법은 기존의 DOS 3.3과 다를 것이 없다. 먼저 램디스크를 실행시킨 후에 "CATALOG, S3, D1"을 입력하면 화면상에 "DISK VOLUME 001"이 나타날 것이다. DOS 3.3과는 달리 램디스크에는 INIT 명령이 수행되지 않는다 (이 램디스크를 이용해서 열심히 프로그램을 제작하고 디스크에 세이브하지 않고 전원을 꺼버린다면 큰 실수를 했음을 깨달을 것이다).

디스크 드라이브에는 읽고 쓰고 할 때마다 점멸하는 LED램프가 있다. 램디스크에도 이러한 것이 있다. "CATALOG"를 하거나 "SAVE" 나 "LOAD"를 할 때 모니터의 오른쪽 하단에 무엇인가 깜빡임을 볼 수 있을 것이다. 좀더 자세히 보면 "LOAD"할 때마다 'R'자가 나타남을 알 수 있을 것이다. 또한 'SAVE' 할 때마다 'W'자가 표시된다.

(2) 램디스크의 포맷

램디스크에서는 INIT 명령이 통하지 않는다. 그러나 램디스크의 파일을 모두 지워야 할 때, 즉 포맷시켜야 할 때는 어떻게 해야 할까. 가장 간단한 방법은 Ⅱe의 전원을 ON-OFF시키는 것이다. 그러나 컴퓨터를 조금이라도 아끼고 이해하는 사람은 이 방법이 아닌 다른 방법을 찾으려 할 것이다. 그래서 또 하나의 방법은 먼저 'BRUN RAMDRIVE, S6, D1'을 친 후에 Solid-Apple()키와 리턴키를 동시에 입력한 후 'CATALOG, S3, D1'을 입력해보자. 모니터상에는 'DISK VOLUME 001'외에는 아무 것도 없을 것이다.

(3) 램디스크의 파라메터 조정

파라메터를 조정하려면 먼저 램디스크 모드로 진입한 후 새로운 파라메터를 정하고 'CALL 24576'을 입력한다.

표6.3은 조정할 파라메터의 번지 목록이다.

80컬럼과 더블 고해상도 그래픽은 램워크의 0번 뱅크에 귀속되어 있다. 따라서 램디스크와도 관련이 있다. 램디스크 프로그램의 24579번지와 24580번지가 80컬럼과 더블 고해상도와 관련된 번지이다. 24579번지는 80컬럼과 관계된 번지로서 0일 때는 80컬럼을 쓰지 않는 경우이고 1일 때는 80컬럼을 쓰는 경우이다. 24580번지는 더블 고해상도 그래픽을 쓰지 않는 경우이고 1일 때는 더블 고해상도 그래픽을 사용하는 경우이다.

만일 보조 메모리가 다른 슬롯으로 옮겨질 때에는 위치를 다시 설정해 주어야 한다. 그곳이 바로 24581번지이다. 만일 보조 메모리가 5번 슬롯에 있다면 POKE 24581, 5를 입력하여야 한다. 또 램디스크의 볼륨 번호를 바꾸고자 한다면 24582 번지의 내용을 바꾸면 된다. 만일 다른 프로그램이 보조 메모리를 사용할 때에는 램디스크가 이곳을 침범하지 말아야 할 것이다. 램디스크는 모두 16개의 뱅크로 이루어져 있다. 이러한 뱅크에 램디스크가 Over Write하지 않으려면 특수한 지정이 필요하다.

예를 들어 0번 뱅크를 Lock시키려면

$2^0 = 1$이 되어 POKE 24598, 1하면 된다.

0과 3번 뱅크를 LOCK하려면

$2^0+2^3 = 1+8=9$로 되어 POKE 24598, 9로 하면 된다.

그런데 8번부터 15번 뱅크까지는 앞의 것과는 조금 다르다.

번지(10진)	초기치(10진)	기 능
24579	1	80 컬럼 모드 설정(0 또는 1)
24580	0	더블 고해상 모드 설정 불가(0 또는 1)
24581	3	보조 메모리의 슬롯 번호 설정(1~7)
24582	1	볼륨 번호 설정(1~254)
24583	24	오디오 인디케이터의 지속 시간을 읽는다(0~255).
24584	18	오디오 인디케이터의 지속 시간을 쓴다(0~255).
24585	6	오디오 인디케이터의 주파수를 읽는다(0~255).
24586	18	오디오 인디케이터의 주파수를 쓴다(0~255).
24587	18	비주얼 인디케이터의 특성을 읽는다(R).
24588	23	비주얼 인디케이터의 특성을 쓴다(W).
24589	0	최하위 드라이브 번호(0~4)
24591	15	드라이브 1에 디렉토리 입력을 7로(1~15)
⋮	⋮	⋮
24596	15	드라이브 6에 디렉토리 입력을 7로(1~15)
24597	0	8~15번 뱅크를 닫는다.
24598	0	0~7번 뱅크를 닫는다.

표 6.3 파라메터의 번지 목록

예를 들어 8번과 9번을 LOCK 하려면
$(2^8+2^9)/256 = (256+512)/256=3$이 되어 POKE 24597,3 하면 된다.

§ 2. 한글 워드프로세서 "작은별" V1.1

우리나라에 컴퓨터가 처음 들어온 이래로 많은 한글 워드프로세서들이 개발되었다. 그러나 기존에 나왔던 중앙한글과 그밖의 한글 워드프로세서들은 한 화면에 영문이 40×24자를 쓰는 데 비해 한글은 영문의 ¼인 20×12자밖에 못써서 불편했었다. 하지만 애플Ⅱe가 이를 해결해 주었다. 더블 고해상도 그래픽(double high-resolution graphic)으로 한글이 한 화면에 40×12자로 두 배 늘어났기 때문이다.

따라서 이 한글 워드프로세서 "작은별"은 많은 양을 처리할 수 있다.

1. "작은별"을 쓰기 위해 필요한 것들

① 애플 Ⅱe, Ⅱc
② 디스크 드라이브 한 대 이상
③ 프린터

2. "작은별"의 장점

① 한글, 영문 40×12자.
② 커서 이동을 사용하기가 쉽다.
③ 에디터(editor) 기능이 강력하다.
④ 사용 가능한 글자의 수가 12,107자로 매우 많은 양을 사용할 수 있다.
⑤ 처리 속도가 빠르다.
⑥ 특수 기호가 많다.

3. 메인 메뉴

화살표키로 이동시켜 리턴키로 선택하거나 번호를 눌러준다.
① 문서 작성/수정
뒤에서 설명하겠다.
② 디스크 목록
CATALOG 명령을 수행한다.
③ 디스크에서 가져옴
문서 작성할 파일을 데이타 디스크에서 가져온다. 가장 최근에 입력한 파일 이름이 기억되어 계속 사용할 수 있다. 파일 이름을 입력할 때 가장 최근에 입력한 것을 사용하려면 Y를 누르고, 그렇지 않으면 아무 키나 누른 후 원하는 파일 이름을 입력한다. Caps Lock키는 대/소문자 변환에 사용한다.
④ 디스크에 저장
작성한 문서를 데이타 디스크에 저장시킨다. 단, DOS 3.3으로 포매팅된 디스크여야 한다. 즉 ProDOS에서는 사용할 수 없다.
⑤ 인쇄
프린터로 출력한다. "작은별"이 사용할 수 있는 프린터는 엡슨(Epson)계열인 FX-80, RX-80, LX-

80, LX-86, FT-5002와 스타(Star) 계열인 GEMINI, DELTA 그리고 금성 PT-80T이다. 각 프린터마다 사용할 수 있는 기능이 정해져 있으므로 주의해야 한다.

⑥ 데이타 드라이브 지정

데이타 드라이브의 슬롯은 6이고, 한 번 수행할 때마다 1, 2를 반복한다.

⑦ 문서 합침

메모리에 있는 문서(내용)에 다른 파일의 내용을 합친다.

⑧ 메모리를 지움

메모리를 지운다. 즉 버퍼에 있는 내용을 지운다.

"메모리를 지웁니까 ? (Y/N)"라는 질문에 Y를 누르면 메모리를 지우고 그렇지 않으면 아무 키나 눌러 빠져나온다.

⑨ 끝냄

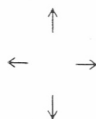
작업을 끝낸다.

"빠져나갑니까 ? (Y/N)"라는 질문이 나오는데 Y를 입력하면 프로그램에서 빠져나와 애플소프트 프롬프트로 돌아간다.

4. 편집 명령키

CTRL(control)을 간단히 "^"로 표시하겠다.

(1) 커서 이동키



① 한 칸 이동 ^E

^S ^D

^X

② 한 단어 이동 ^A ^F

③ 한 화면 이동 ^R

^C

먼저 CTRL+Q를 누른 후 다음의 해당키들을 누른다.

④ 문장 이동 ^[J] ^[K]

⑤ 행 이동 ^[S] ^[D]

⑥ 화면 이동 ^[E] ^[X]

⑦ 파일 이동 ^[R] ^[C]

(2) 삽입과 삭제

^V : 삽입과 겹침 전환

^N : 새로운 행 삽입

DEL : 앞문자를 지우고 한 칸씩 당겨진다.

^H : ←키를 누른 것과 같으며 DEL키와 같은 기능을 한다.

^G : DEL키와 비슷하지만 앞문자가 아닌 커서 위의 한 문자를 지우고 한 칸씩 당겨진다.

^T : 커서 오른쪽의 단어 하나를 삭제한다.

^Y : 문장 하나를 삭제한다.

여기서부터는 ^Q를 누른 후 다음의 해당키들을 누른다.

^H : 행 하나를 삭제한다.

^Y : 커서에서부터 문장 끝까지 삭제한다.

^N : 커서 뒤에 두 줄의 공백을 넣어준다.

^G : 커서 뒤의 공백을 없애준다.

㈜ ^Q ^N과 ^Q ^G는 삽입할 때 속도가 늦어지는 것을 보완해 주는 명령키로, ^Q ^N을 누른 후 뒤의 공백에 겹침 모드로 입력하고 ^Q ^G를 눌러 공백을 없앤다.

(3) 치환

^Q ^A : "검색 : "에 바꾸고자 하는 문자를 입력하고, "치환 : "에 치환할 문자를 입력한다. 그러면 "모두 치환합니까 ? (Y/N)"가 나오는데 Y를 누르면 모두 치환하고, N을 누르면 문자를 찾아 치환할지 여부를 물어 원하는 것만 치환할 수 있다.

(4) 검색

^Q ^F: "검색 : "에 찾고자 하는 문자를 입력하면 찾아서 커서를 위치시켜 준다.

^L : ^Q ^F로 입력한 문자를 계속 찾는다.

㈜ 치환과 검색은 커서가 있는 곳부터 찾으므로 파일 전체에 사용하려면 ^Q ^R로 이동한 후에 한다.

(5) 블럭 명령

원하는 부분을 지정해서 복사, 이동 등을 할 수 있다. 먼저 ^K를 누르고 다음의 해당키들을 누른다.

^B : 블럭의 처음을 지정한다.

^K : 블럭의 끝을 지정한다.

^C : 블럭을 복사한다.

^V : 블럭을 이동한다.

^Y : 블럭을 삭제한다.

^H : 블럭의 표시 ON/OFF.

^W : 블럭을 디스크에 써넣는다.

^R : 블럭을 디스크에서 읽는다.

(6) 그밖의 명령들

^J : 한글 상태에서 오타가 발생하였을 때 이 키를 누르면 현 커서의 위치에서 새로 입력할 수 있도록 해준다.

^O ^W : 커서 위치와 커서 다음의 문자를 서로 교환한다. 즉 "MY"라고 입력할 것을 빨리 입력하다가 "YM"으로 입력하였을 경우 "Y"에 커서를 위치하고 ^O ^W를 압력하면 "MY"로 바꿀 수 있다.

^O ^Y : 커서 위치의 문자가 영문일 경우 대문자는 소문자로, 소문자는 대문자로 각각 바꾼다. 한글의 경우는 변동없다.

^U : 직전에 지운 내용을 복구한다. 이 기능은 또한 Cut & Paste 기능으로 이용될 수 있다. 즉 원하는 내용을 삭제하면 그 내용이 버퍼에 저장되고 이 버퍼의 내용은 ^O ^C로 지우거나 다른 내용이 삭제되어 버퍼로 들어올 때까지 보존되므로 ^U로 여러번 이용할 수 있다.

^O ^C : 버퍼의 내용을 지운다.

^O ^G : 키를 누를 때마다 나는 키클릭음을 ON/OFF한다.

^K ^D : 편집 모드에서 빠져나가 메뉴로 나간다.

^P : 컨트롤 문자를 입력한다. 이 기능을 이용하여 프린터에 맞는 제어 명령을 프린터에 보낼 수가 있다. 단 그 기능은 그 프린터 고유의 기능이 아니라 한글 모듈에 내장된 기능이어야 한다.

ESC : 키보드 버퍼에 들어있는 내용을 모두 지운다.

5. 프린트에 대해서

프린터에 출력을 하기 위해서는 주메뉴에서 인쇄를 선택하면 된다. 그런데 이 경우는 프린터의 한글 모듈을 이용하여 프린팅하기 때문에 속도는 빠르지만 특수 문자들을 프린트할 수 없다. 따라서 특수

문자들을 프린트하기 위해서는 "9. 끝냄"을 선택하여 일단 "작은별"에서 빠져나가야 한다. 단 빠져나가기 전에 작업하던 문서는 디스크에 반드시 저장해 두어야 한다. 일단 "작은별"에서 빠져나오면 RUN PRINTER를 입력한다.

6. 파일 변환에 대하여

"작은별"은 스마텀 카드용의 한글 워드프로세서인 "스마트 워드"와는 데이타 호환성이 있으며, "중앙한글"의 데이타는 변환 프로그램인 CONVERT로 "작은별"에 맞게 바꿀 수 있다. 이때 중앙한글에서 USER 모드를 사용하여 입력한 특수 문자들은 모두 공백으로 바뀐다.

7. 특수키

Solid-Apple : 한글과 영문 전환

Open -Apple : 다른 키와 함께 누르면 프린터 제어 문자와 특수 문자를 입력할 수 있다.

(참고로 "ㄲ, ㄸ, ㅃ, ㅆ, ㅉ, ㅒ, ㅖ"는 Caps Lock키를 누르고 "ㄱ, ㄷ, ㅂ, ㅅ, ㅈ, ㅐ, ㅔ"를 누르면 된다.

8. 특수 문자

```
! -            E - Σ            b - ㄴ
" -            F - σ            c - ㄱ
# -            G - μ            d - ┘
$ -            H - τ            e - —
% -            I - Φ            f - |
& -                    J - Θ
' - エ
, -                                        K - Ω
          h - ⊥
( -
```

9. 특수 문자 정리

특수 문자 정리

1	2	3	4	5	6	7	8	9	:
Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ

```
U  ╱         ┌─┬─┐   aegec
V  ╱         │ │ │   f f f
             ├─┼─┤   iekej
             │ │ │   f f f
             └─┴─┘   behed
```

L — §

i — ⊢

1 — Ⅰ		N — ∅	
	k — 十		
2 — Ⅱ		O — ∈	
	l — 日		
3 — Ⅲ		P — ∩	
	m — 月		
4 — Ⅳ		Q — ≡	
	n — 火		
5 — Ⅴ		R — ±	
	o — 水		
6 — Ⅵ		S — ≥	
	p — 木		
7 — Ⅶ		T — ≤	
	q — 金		
8 — Ⅷ		U — ⌠	
	r — 土		
9 — Ⅸ		V — ⌡	
	s — 年		
: — Ⅹ		W — ÷	
	t — ☎		
; — ½		X — ≈	
	u — ~		
< — ⅓		Y — °	
	v — →		
= — ⅔		Z — °	
	w — ←		
> — ¼		[— •	
	x — ↔		
? — ²⁄₄		\ — √	
	y — ∀		
@ — ¾		] — ⁿ	
	z — ≡		
A — α		^ — ²	
	{ — ♥		
B — β		_ — ↵	
	| — ♦		
C — Γ		` — ₩	
	} — ♣		
D — Π		a — ┌	
	~ — ♠		
DEL —			

제 7 장

애플의 표준 OS인 ProDOS 및 ProDOS BASIC

§1. ProDOS의 기본적인 면

§2. ProDOS의 특징적인 명령들과 에러 메시지

애플Ⅰ이 나온 이래로 컴퓨터들의 개선된 기능과 그 동안 계속되어 온 하드웨어 발전을 수용할 수 있는 DOS(Disk Operating System)가 필요하게 되었다. 이에 따라 애플사에서는 기존의 DOS3.3을 개량하지 않고 아주 새로운 DOS인 ProDOS(Professional DOS)를 발표하였다. 이 ProDOS에는 자체의 BASIC 인터프리터가 있어서 BASIC상에서 내려지는 ProDOS 명령들을 처리한다. ProDOS로 작성된 프로그램들이 많이 쏟아져 나오는 요즈음 이 ProDOS BASIC을 필히 알아두어야 할 것이기에 자세히 설명하고자 있다.

§1. ProDOS의 기본적인 면

1. ProDOS BASIC을 사용하기 위하여 필요한 조건

• 64K 애플소프트 베이직 롬을 갖춘 애플Ⅱ, Ⅱ+ , Ⅱe, Ⅱc (여기서 문제가 되는 것이 애플소프트 롬이다. 왜냐하면 ProDOS는 표준 애플소프트 롬이 있어야 동작을 하기 때문이다. 즉 전원을 켰을 때 화면 상단 가운데에 "Apple Ⅱ"라고 표시되거나 한글이 내장되어 있지 않아야 한다. "Apple Ⅱ"라고 표시되지 않을 때에는 롬을 교체해 주거나 ProDOS를 수정해야 한다.)

• 한 대 이상의 디스크 드라이브

• 이 밖에 클럭카드, 하드디스크 드라이브, 80컬럼 카드 등이 있으면 더욱 좋다.

• ProDOS에는 1.0, 1.0A, 1.0.1, 1.0.2, 1.1.1 등 여러 버전이 있는데 별로 상관할 것이 없다. 다만 1.0은 DOS와 인터프리터에 문제가 있으므로 사용하지 말기 바란다. 1.0A는 ProDOS가 작동되지 않는 호환 기종을 위해 수정된 것으로, 이 버전이 들어있는 AppleWorks 디스크를 부팅시키고 프롬프트가 나오면 리셋을 걸어서 다른 프로그램을 사용하고 이것마저 안되면 롬을 교체하는 것이 좋겠다.

2. Ⅱe에서의 ProDOS 배치도

다음 그림은 애플(64K)상의 ProDOS가 위치해 있을 때의 메모리 맵이다(128K의 Ⅱe도 이와 같다. 단 Aux. Memory의 64K를 / RAM으로 쓴다).

3. ProDOS의 구조

보통 상태에서 ProDOS는 뱅크 전환 메모리 16K와 메인 메모리 10K 정도를 차지한다.(그림7.1 참조) 여기서 16K에 위치하는 MLI(기계어 인터페이스)라는 부분이 있는데 이 루틴은 어떠한 기계어 프로그램이라도 디스크 억세스를 가능하게 한다. 그리고 메인 메모리에 위치하는 베이직 시스템은 일종

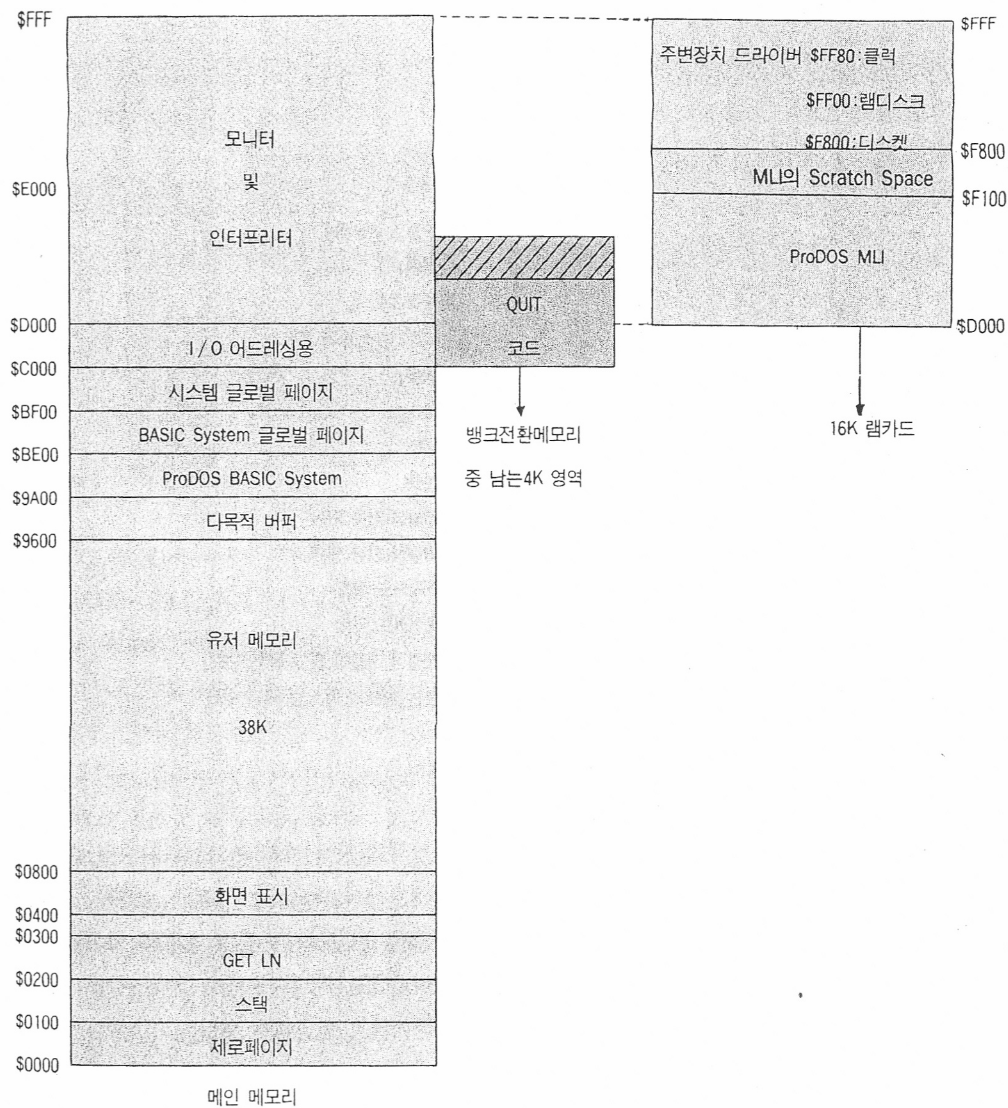


그림 7.1 ProDOS의 메모리 배치도

의 인터프리터 모듈이라 볼 수 있으며 파스칼 언어를 처리하는 모듈이 설치될 수도 있고 자기 취향에 맞는 인터프리터를 위치시킬 수 있다. 하지만 현재는 애플소프트용밖에는 없다. 그리고 이 베이직 시스템이 로드되면 HIMEM은 자동적으로 그 아래로 세트된다. 또 MLI와 베이직 시스템의 효율적인 운영을 위해 두 개의 글로벌 페이지(global page)가 있어서 관련되는 어드레스와 데이타들을 담고 있다.

예를 들어 시스템 글로벌 페이지의 MACHID(MACHine ID byte)라 불리는 $BF98에 기록된 수치

값에 따라 현재 사용하는 시스템의 종류를 체크하게 된다.

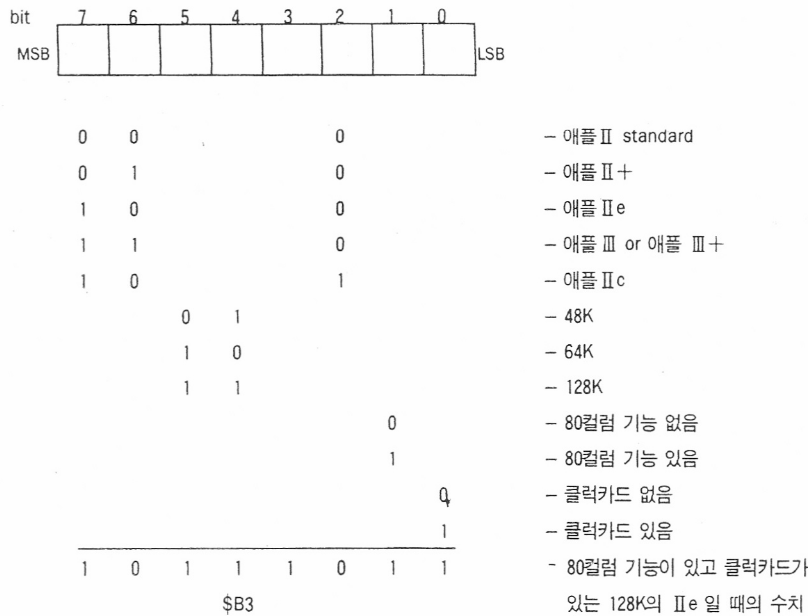


그림 7.2 MACHID DATA

베이직 시스템이 위치한 영역 밑에서 다목적 버퍼라는 것이 있고 이 다목적 버퍼 사이에 파일 버퍼라는 것이 존재한다. 이 파일 버퍼의 크기에 따라서 유저 메모리의 변동이 생기는데 오픈된 파일 수가 많을수록 유저 메모리는 줄어든다. ProDOS MLI는 뱅크 전환 메모리의 16K중 12K를 차지하며 남는 4K의 어느 정도는 QUIT코드가 설치되어 있다. 그리고 ProDOS는 주변장치들을 위한 인터럽트도 처리하는데 MLI와 Scratch Space 위쪽 2K가 주변장치 드라이버로 할당되어 있다.

디스크의 구성을 살펴보면 ProDOS도 트랙/섹터 개념을 적용한다는 점 이외에는 DOS 3.3과는 완전히 다르다. 애플용의 플로피디스크 드라이브는 트랙/섹터 형식으로 기록이 되는데 이것을 가동시키는 드라이버 프로그램이 트랙/섹터 블럭 사이에서 매핑(mapping)을 하게 된다. 쉽게 말하자면 어느 트랙/섹터가 어느 블럭인가를 판단하여 억세스를 돕는 것이다. 한 섹터에는 256바이트가 기록되므로 한 블럭에는 두 섹터가 필요하고 한 장의 디스크에는 이런 형식으로 280개의 블럭이 형성되는 것이다.

다음으로 디스크의 블럭 배치를 알아보기 위해서 시스템 디스크를 예로 들어 살펴보겠다.
설명을 읽어나가면서 여러분이 직접 살펴보면 좋을 것이다. 시스템 디스크의 디스크 볼륨에서는 처음 일곱 개 블럭이(0~6블럭) 시스템의 기본 구조에 할당되어 있고 다음 세 가지가 기록된다.

트랙 수	35
트랙당 섹터 수	16
디스크당 섹터 수	560
섹터당 기록 바이트 수	256
디스크당 기록 바이트수	143,360
데이타 저장에 사용가능한 블럭 수	
ProDOS 시스템 디스크	221
ProDOS 데이터 디스크	273
디스크당 기록 가능한 바이트 수	
ProDOS 시스템 디스크	113,152
ProDOS 데이타 디스크	139,776

표 7.1 디스크 구성

① 부트스트랩 로더(0~1 블럭)

부팅시킬 때 처음으로 읽혀지는 부분으로 부팅을 가능하게 해준다.

② 볼륨 디렉토리(2~5블럭)

DOS 3.3의 카탈로그 트랙과 유사한 부분으로 첫번째 두 블럭은 키 블럭으로서 볼륨 전체에 대한 정보를 담고 있는 헤더가 첫 엔트리로 자리잡고 있다. 그리고 등록 가능한 파일 수는 51개이고 시스템 디스크는 세 개의 파일이 등록되어 있을 것이다. 각 블럭의 첫 2바이트는 앞블럭의 번호, 다음 2바이트는 다음 블럭의 번호를 표시하며, 헤더의 첫 바이트는 다음 블럭/앞블럭을 가리키는 블럭 포인터가 첫 4바이트를 차지하므로 + $04에서 시작된다.

· $TORAGE-TYPE / NAME-LENGTH($04)

이 바이트의 상위 4비트는 엔트리의 종류를 표시하며 시스템 디스크에는 볼륨 디렉토리의 헤더임을 표시하는 $F 값이 들어있다.

· VOLUME-NAME($05~$13)

볼륨의 이름을 담고 있는 15바이트 길이의 필드이다.

· CREATION($1C~$1F)

$14~$1B-미사용

이 볼륨이 만들어진 날짜와 시간을 저장한다.

· VERSION($20)

이 볼륨에 포매팅된 ProDOS의 버전 번호로 현재는 $0가 설정되어 있다.

· MIN-VERSION($21)

이 볼륨이 억세스할 수 있는 ProDOS의 최소 버전을 표시한다.

· ACCESS($22)
 일종의 플래그 바이트이다.
· ENTRY-LENGTH($23)
 볼륨 디렉토리의 각 엔트리의 길이이다.
· ENTRIES-PER-BLOCK($24)
 볼륨 디렉토리의 각 블럭에 등록되는 엔트리의 수로 거의 $0D로 설정되어 있다.
· FILE COUNT($25~$26)
 사용 가능한 엔트리의 수이다.
· BIT-MAP -POINTER($27~$28)
 볼륨 비트맵이 있는 블럭의 시작 번호이다. 거의 6으로 설정되어 있다.
· TOTAL-BLOCKS ($29~$2A)
 이 볼륨의 총 블럭수 35TRACK 디스크에서는 $0118(280_{10})이 설정되어 있다.
 ③ 볼륨 비트맵(6블럭)
 파일의 수록 상태를 나타내며 비트맵의 크기는 ProDOS가 자동적으로 판단을 내려 할당해 준다.

4. ProDOS의 파일 구조

이 책의 앞부분에서 잠시 다룬 적이 있는데, 여기서는 좀더 자세히 살펴보자.

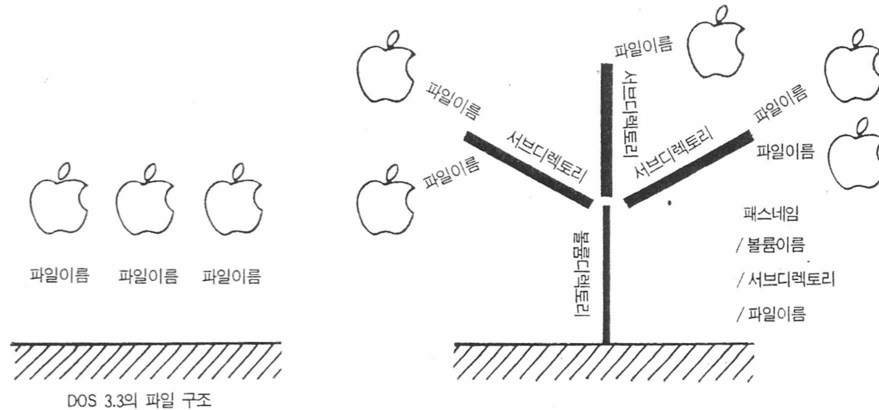


그림7.3 ProDOS 파일의 트리 구조

그림7.3에서와 같이 ProDOS는 계층적 구조(hierarchical structure)로 파일을 구성한다. 즉 트리 모양으로 구조가 형성된다.

ProDOS의 파일 구조는 뻗는 가지와 같이 볼륨 디렉토리로부터 성장한 서브디렉토리, 또 그로부터 성장한 파일 이름으로 되어 있다. 따라서 ProDOS 파일을 찾으려면 볼륨 디렉토리를 따라 서브디렉토리, 파일 이름을 찾아야 한다. 이 /볼륨 이름/서브디렉토리/파일 이름을 Pathname이라 하는데 자세한 것은 뒤에서 다루겠다.

예를 들어 프로그램 리스트에 관한 정보를 저장하는 ProDOS 디스크를 /PROGRAM이라는 이름으로 사용한다고 하자. 그림7.4와 같이 /GAME과 /UTILITY라는 종류에 나누어 저장할 수 있다.

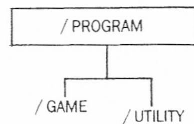


그림7.4 레벨1의 ProDOS 디렉토리

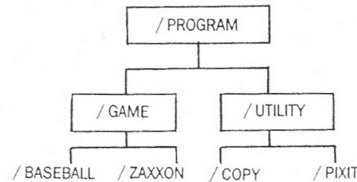
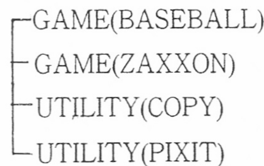


그림7.5 레벨2의 ProDOS 디렉토리

그러나 자세하게 그 제목까지 저장한다고 하면 그림7.5와 같이 서브디렉토리로 구성할 수 있다.



위의 그림들을 보면 DOS 3.3의 파일 구조와 ProDOS 파일 구조를 비교할 수 있을 것이다. 그리고 ProDOS의 트리 구조는 하드디스크 사용시의 이점이 있다. DOS 3.3의 레벨 1의 파일 배열은 플로피디스크와 같은 적은 저장 용량에 적합하고 수천 개의 파일을 가지는 하드디스크는 서브디렉토리의 이름에 의해서 하드디스크를 여러 그룹의 플로피디스크 크기 정도로 변화시켜 사용하므로 한층 편리하다.

(1) Pathname(패스네임)

ProDOS에서 파일을 정의하기 위해서는 볼륨 디렉토리, 서브디렉토리, 파일 이름을 써주어야 하는데 이런 연속된 명칭을 패스네임이라 부른다. 이 패스네임의 규칙은 다음과 같다.

① 볼륨 디렉토리, 서브디렉토리, 파일 이름을 포함('1'도 포함)한 문자수는 64개 이하여야 한다.

② 형식 : /볼륨 디렉토리/ 서브디렉토리/ 파일 이름

③ 각각의 볼륨 디렉토리, 서브디렉토리, 파일 이름은 15문자 이하여야 한다.

④ 각각의 볼륨 디렉토리, 서브디렉토리, 파일 이름의 첫문자는 반드시 알파벳이어야 하고 나머지 문

자는 알파벳, 숫자, 점이어야 한다. 스페이스도 허용이 안된다.

(2) 파일 크기에 따른 관리

ProDOS는 파일의 크기에 따라서 다음과 같이 세 가지로 판단, 분류하여 취급한다.

Seeding File(길이 512B 이하인 파일)

Sapling File(길이 512B~128KB인 파일)

Tree File(길이 128KB 이상인 파일)

이러한 파일 종류는 볼륨 비트맵 결정시 영향을 준다.

5. ProDOS BASIC DISK 만들기

COPYⅡ PLUS 6.3을 이용한다. 이 프로그램은 센트럴포인트 소프트웨어사에서 계속해서 발표한 것중 하나이다. 이 버전은 DOS 3.3과 ProDOS를 같이 사용할 수 있어 매우 좋다. 또 ProDOS USER'S DISK와 ProDOS 유틸리티 디스크의 모든 기능을 가지고 있으므로 속도도 느린 위의 두 프로그램을 사용할 필요가 없다. 그리고 COPYⅡ PLUS 6.3은 DOS 3.3에서 사용하던 4.3이나 5.1과 사용법이 거의 같으므로 사용하기도 쉽다. Bit Copy의 분석 능력 향상과 데이타의 저장 상태를 보여주는 Hi-res Scanning 기능까지 사용할 수 있다. ProDOS와 DOS 3.3의 파일 변환은 Copy기능에서 프로그램이 자동적으로 ProDOS로 포매팅됐는지, DOS 3.3으로 포매팅됐는지를 검사하여 그것에 맞게 읽거나 쓰기 때문에 ProDOS이든 DOS 3.3이든 상관없이 파일 변환이 쉽게 된다. 한 가지 주의할 점은 위와 같이 파일을 변환했을 때 변환된 프로그램이 실행되지 않을 수도 있다. 왜냐하면 ProDOS와 DOS 3.3은 파일의 이름부터 완전히 다른 구조로 되어있기 때문이다. 이에 대해서는 ProDOS BASIC을 설명하는 가운데 다루도록 하겠다.

ProDOS BASIC DISK를 만들려면 다음과 같은 순서로 한다.

① COPYⅡ PLUS 6.3을 부팅시킨 후 날짜를 입력한다. 입력 안해도 상관없다 . 80컬럼 카드를 사용하겠느냐는 질문에 Y나 N을 입력한다. 이제 카피할 준비가 되었다.

② 공디스크를 넣고 포매팅을 한다.

③ ProDOS와 BASIC. SYSTEM 두 파일을 카피 기능을 이용해 카피한다. 만약 디스크 드라이브가 한 대라면 램디스크를 이용해도 괜찮을 것이다. 먼저 FORMAT DISK를 선택한 다음 ProDOS를 선택하면 SELECT DEVICE가 나오는데 여기서 맨 위의 SLOT 3 DRIVE 2 : /RAM DISCONNECTED 를 선택한다. 그 다음에 Y를 누르면 된다. 즉시 포매팅이 되고 리턴을 누르면 된다. 여기서 다시 COPY를 선택한 후 FILES를 선택한다. 드라이브와 ProDOS와 BASIC SYSTEM 두 파일이 들어있는 디스크를 넣고 SLOT 6 DRIVE 1을 선택한 후 SLOT 3 DRIVE 2를 선택하면 화면에 카탈로그

가 나온다. 여기서 두 파일을 선택하고 G를 눌러 카피한다. 다시 COPY를 선택한 후 FILES를 선택하고 포매팅한 디스크를 넣는다. 이번에는 위와는 반대로 DEVICE를 선택한다. 그러면 화면에 두 파일이 나올 것이다. 역시 두 파일을 선택한 후 G를 눌러 카피한다. 이렇게 하면 디스크를 세 번만 넣었다 빼면 되므로 간편하다. 설명이 길어서 어렵게 보일지는 모르나 해보면 쉽다는 것을 느낄 것이다. 위에서 세 번만 넣었다 빼면 된다고 했는데, 현명한 독자라면 두 번에 할 수 있을 것이다. QUIT를 선택해서 리셋을 누르면 부팅이 된다. STARTUP 프로그램이 없으므로 아래와 같이 메시지가 나오고 프롬프트가 나온다.

```
PRODOS BASIC 1.0
COPYRIGHT APPLE, 1983
```

이제부터 ProDOS BASIC을 쓸 수 있게 되었다. 하나하나 설명하기로 한다.

6. ProDOS BASIC의 장단점

(1) ProDOS BASIC의 장점

① BASIC 프로그램 작성을 위해 새로운 기능이 추가되었다. BASIC에서 디렉토리 파일(DIR)을 읽을 수 있고, 변수들도 디스크에 저장했다 복구할 수 있으며, 변수들을 보존하면서 프로그램을 연결시킬 수 있다.

② 애플소프트의 스트링에 대한 가베지 컬렉션(garbagy collection)이 전보다 몇 배 더 빠르고 효율적이다.

③ 랜덤억세스 텍스트 파일의 레코드 길이가 파일에 기록되므로 레코드의 길이를 몰라도 BASIC 프로그램이 억세스할 수 있다.

④ 디렉토리에 파일에 대한 정보가 더 많이 들어간다. 예를 들면, 애플소프트나 기계어 파일의 길이가 파일 자체에 들어가는 것이 아니고 디렉토리에 들어간다.

⑤ ProDOS의 BASIC 명령 처리 프로그램이 BASIC 프로그램의 명령 라인을 인터셉트하는 방법이 개선되고 신뢰도가 향상되었다. 따라서 ProDOS와의 연결이 끊어지는 경우가 DOS 3.3에 비해서 크게 줄어들었다.

(2) ProDOS BASIC의 단점

앞에서와 같은 장점이 있는 반면에 단점도 있다.

① DOS 3.3에서 만들어진 대부분의 프로그램(BASIC, 어셈블리어,…)은 다시 작성하거나 수정해야

한다.

② 정수 BASIC을 쓸 수 없다. 메인보드의 롬에 애플소프트가 있어야 ProDOS BASIC 명령 처리가 제대로 동작하기 때문이다.

③ BASIC 프로그램 작성과 명령어 사용에는 반드시 64K가 필요하다. 모든 ProDOS 명령을 실행시키는 BASIC. SYSTEM이 작동하려면 64K가 필요하기 때문이다.

④ BASIC 프로그램을 작성할 때 이용할 수 있는 메모리 용량이 줄어든다. DOS 3.3에서는 세 개의 파일 버퍼를 설정하면(디폴트) HIMEM이 $9600에 세트되지만, ProDOS에서는 파일 버퍼가 없는 상태에서 $9600에 세트된다. 따라서 파일을 OPEN할 때마다 HIMEM이 1K씩 내려간다. 또 DOS 3.3은 DOS를 램카드에 이동시켜 BASIC 메모리 용량이 늘어나고 DOS 아래쪽의 4K 뱅크에 애플소프트의 기능 보강 프로그램도 넣을 수 있지만, ProDOS 시스템 파일이 16K 램카드나 뱅크 전환 메모리에 위치하고 Quit코드가 4K 뱅크에 위치하는 ProDOS에서는 사용할 수가 없다.

⑤ DOS 3.3에서는 동시에 16개의 파일을 OPEN할 수 있었지만, ProDOS에서는 여덟 개까지만 가능하다. 한 개의 OPEN된 파일이 차지하는 버퍼가 DOS 3.3에서는 595바이트이지만, ProDOS에서는 1024바이트나 된다.

⑥ 계산 위주의 프로그램은 DOS 3.3보다 약 4% 느려진다. 이것은 ProDOS BASIC 명령 처리 프로그램이 실행 상태를 추적하고 디스크 명령과 가베지 컬렉션을 수행하기 위해 애플소프트의 TRACE 기능을 작동 상태로 두기 때문이다. 그러나 스트링이나 디스크 억세스가 있을 경우에는 이에 따른 기능 향상이 앞의 단점을 상쇄하고도 남는다.

⑦ NOMON, MON, VERIFY 등 몇 가지 명령이 삭제되었다. 따라서 EXEC 파일이 실행될 때 명령들을 볼 수 없게 되었다.

(3) 기타 특징

① 기계어 프로그램을 BRUN시키면 DOS 3.3에서는 해당 프로그램으로 점프가 되지만, ProDOS는 그것을 CALL한다. 따라서 리턴 서브루틴에 의해 ProDOS로 돌아올 수 있다.

② 지정한 이름의 파일이 현재 오픈되어 있지 않아도 CLOSE 명령이 에러가 되지 않는다.

③ APPEND에 WRITE 기능이 들어있어서 APPEND 뒤에 꼭 WRITE명령을 할 필요가 없다.

④ 참고 사항으로 디렉토리의 내용중에 ASCII 텍스트나 텍스트 파일은 모두 최상위 비트가 Off되어 기록된다.

⑤ BASIC. SYSTEM에 의해 ProDOS BASIC이 작동하는데 만약 기계어만을 사용할 경우에는 이것이 필요없게 된다.(예 COPY Ⅱ PLUS). 실제로 ProDOS 시스템 파일이 로드되어 ProDOS가 자리를 잡으면 이름이 SYSTEM으로 끝나는 프로그램을 찾아 실행시키고 그 다음 STARTUP을 실행시킨다. 따라서 기계어만을 사용할 경우 이름 뒤에 .SYSTEM을 붙여서 ProDOS 파일 바로 다음에 수

록하면 된다.

7. ProDOS BASIC Programming Examples 디스크

ProDOS BASIC Programming Examples 디스크를 이용해서 ProDOS BASIC을 배우면 훨씬 쉽게 이해할 수 있다. 이 디스크의 설명서인 BASIC Programming With ProDOS 매뉴얼은 이 디스크를 이용한 자습서(tutorial) 형식으로 되어 있다.

특히 이 디스크에는 다양한 샘플 프로그램이나 유틸리티 외에도, HELP 프로그램이 있어서 작동시켜 두면 프로그램 작성 도중에도 명령어들에 대한 도움말을 얻을 수가 있다. 이것은 일종의 RAM상주 프로그램으로

– /EXAMPLES/HELP

하면 메모리로 들어와서 다른 프로그램이 사용되기 전까지는 그 기능을 유지한다. 도움말을 요청하는 방법은 세 가지가 있다.

① HELP

이용 가능한 명령어들의 리스트를 보여준다.

② HELP HELP : Help기능 자체
SYNTAX : 명령어의 구문 형식
BINARY : 2진 파일
FILE : 파일 형태

③ HELP [원하는 명령어]

원하는 명령어의 기능, 형식, 사용예, 설명이 나온다.

HELP 화면 우측 상단의 〈IMM & DEF〉에서 IMM은 키보드로부터 한 명령어를 입력하여 직접 실행시키는 직접 모드(immediate mode)에서 사용, DEF는 프로그램의 한 라인으로 실행시키는 참조 모드(deferred mode)에서 사용할 수 있음을 나타낸다.

이 HELP 기능의 정지는 NOHELP이다.

8. 삭제된 명령

여섯 개의 DOS 3.3 명령이 삭제되었다.

① FP와 INT

ProDOS는 애플소프트 BASIC만을 수용하므로 필요가 없게 되었다. 사용하면 ?SYNTAX ERROR가 난다.

② INIT(Initialize)

DOS 3.3에서는 자체에 디스크를 초기화하는 명령이 있었지만, ProDOS는 이 명령을 없애고 ProDOS USER'S DISK의 ProDOS Filer와 같이 포매팅 기능이 들어있는 프로그램으로 공디스크를 포매팅시킨 다음 시스템 파일 두 개를 카피하고 DOS3.3에서의 그리팅(greeting) 프로그램에 해당하는 STARTUP 파일이 있어야 한다. 이 파일의 이름은 꼭 STARTUP이어야 한다. 그 대신에 BASIC, 기계어, EXEC 프로그램 모두 쓸 수 있다. INIT 명령을 쓰면 역시 ?SYNTAX ERROR가 난다.

③ MON, NOMON

디스크 명령과 입출력의 추적 기능은 삭제되었다. MON은 완전히 없어졌으므로 ?SYNTAX ERROR가 나고, NOMON은 에러는 나지 않으나 무시된다.

④ MAXFILES

ProDOS는 한 개의 파일을 OPEN할 때마다 자동적으로 1024바이트의 버퍼를 하나씩 설정해 내려가므로 이 명령은 무시만 되며 에러는 나지 않는다(참고로 64K 애플Ⅱ에서의 베이직 프로그램의 최대 크기는 $A000 – $400n, n은 OPEN된 파일의 총수).

§2. ProDOS의 특징적인 명령들과 에러 메시지

1. 개선되거나 변경된 명령

① APPEND(모든 명령 형식의 괄호 안은 생략 가능)

어떤 파일에도 새로운 데이타를 추가할 수 있으며, 랜덤억세스 파일의 마지막 레코드 바로 다음에 있는 레코드의 첫부분부터 데이타를 추가시킬 수도 있다. 형식은

```
APPEND pathname, Txxx( , Sn) ( , Dn)
```

가. 패스네임은 DOS 3.3의 파일 이름과 유사한 것으로서 ProDOS의 계층적인 트리 구조(tree structure) 때문에 생기는 볼륨 이름과 서브디렉토리의 이름들, 파일 이름들을 합친 것으로 다음과 같은 규칙이 있다.

· '/볼륨 이름/서브디렉토리 이름/파일 이름'과 같은 형태로 되어있다.

· Pathname은 '/'를 포함해서 모두 64자 이하여야 하고 '/'로 구분된 각각의 이름은 15자 이하여야 한다.

예 /GAME/ARCADE/LODE. RUNNER
/GAME/R. P. G./ULTIMA
/GAME/ADVENTURE/TEXT/ZORK
/GAME/ADVENTURE/GRAPHIC/TIME. ZONE

나. Txxx(file type) : 파일의 종류로서 세 자의 약자로 표시된다. 뒤에 표로 알아보기 쉽게 나타내었다.

예 TTXT – 텍스트 파일을 나타낸다.

다. Sn(Slot number) : 슬롯의 번호를 나타낸다.

예 S6

라. Dn(Drive number) : 디스크 드라이브의 번호를 나타낸다.

예 D1

② BLOAD

어떠한 형태의 파일도 쓸 수 있고, 일부만 로드시킬 수 있으며 로드 범위를 지정할 때 시작 번지와 길이 또는 시작 번지와 끝번지로 지정할 수도 있다.

```
BLOAD pathname( , An) ( , Bn) ( , Ln or En) ( , Txxx) ( , Sn) ( , Dn)
```

가. An(Address number) : 프로그램의 시작 번지(예 : A8192 = A$2000)

나. Bn(Byte number) : 파일 내에서 로드를 시작할 첫바이트의 번호이다.

다. Ln(Length number) or En(End number) : Ln은 프로그램의 길이로 DOS 3.3과 같지만, En은 새로운 옵션 파라메터로 프로그램 끝번지를 나타낸다. 따라서 프로그램의 길이를 계산할 필요가 없어졌다. Ln과 En은 둘 중 하나만 쓰거나 둘 다 생략해서 쓸 수 있다(A$2000, L$2000 = A$2000, E$3FFF).

③ BRUN

기계어 파일(2진파일, BIN 형태)을 일부분만 로드시켜 실행시킬 수 있다. 형식은 BLOAD의 형식에서 Txxx를 뺀 것과 같다. 이 BRUN 명령은 기계어 파일을 로드한 후 An(생략되면 원프로그램의 시작 번지)으로 점프(JMP)한다. 이전에 설명한 바와 같이 BRUN은 로드한 후 CALL한다. 따라서 실행했을 때 끝나는 부분이 6502 어셈블리 명령 RTS인 기계어 파일을 BASIC 프로그램의 중간에서 BRUN으로 실행하면 DOS 3.3에서는 BASIC과 연결이 끊어지지만, ProDOS에서는 BASIC과 ProDOS와의

연결을 계속 유지할 수 있다.

④ BSAVE

메모리 내용을 어떤 형태의 파일로도 기록할 수 있다.

BSAVE pathname, An, Ln or En(, Bn) (, Txxx) (, Sn) (, Dn)

* 참고로 BLOAD, BSAVE의 Txxx는 생략하면 TBIN이다.

⑤ CAT & CATALOG

DOS 3.3 보다 더 많은 파일 정보가 표시된다.

가. CAT : 파일의 이름, 형태, 크기, 수정 시기가 포함된 40컬럼 넓이의 카탈로그.

나. CATALOG : CAT에 파일의 작성 시기, 파일 끝의 위치, 기술적인 추가 정보 등이 포함된 80컬럼 넓이의 카탈로그. 그런데 40컬럼 화면에서는 한 파일의 정보가 두 줄로 나오므로 매우 혼란스럽다. 따라서 80컬럼을 사용할 때 쓰는 것이 좋다.

다. 아래에는 사용되지 않고 남은 블럭수, 사용된 블럭수가 표시되며, CATALOG에서는 총 블럭수도 표시된다.

CAT (pathname) (, Sn) (, Dn)
CATALOG (pathname) (, Sn) (, Dn)

· 옵션들이 지정한 디렉토리의 파일들의 리스트를 보여준다. Pathname이 생략되면 현재의 Prefix 디렉토리가 지정되며, Prefix가 비어 있거나 Sn, Dn이 사용되면 볼륨 이름이 지정된다.

라. 각 항목의 내용

· 왼쪽 위에는 디렉토리의 이름이 표시된다. 볼륨 디렉토리일 때에는 앞에 /가 붙는다.

· NAME : 파일명(Lock되면 *가 붙는다.)
· TYPE : 파일 종류를 세 자의 약자로 표시
· BLOCKS : 파일이 차지한 블럭 수
· MODIFIED : 가장 최근에 수정된 날짜, 시간
· CREATED : 파일이 작성된 날짜, 시간
· ENDFILE : 파일에 배정된 디스크 공간을 완전히 채울 경우에 파일의 바이트 수
· SUBTYPE : 파일의 종류에 따라서 몇 가지 추가 정보가 표시된다. 내용은 첫문자에 의해 구분된다.

A… 기계어 파일이 로드되는 어드레스(address). 세이브시킬 때의 시작점이었던 곳이며, 16진수로 표시된다. 예 A = $2000

R… 텍스트 파일이나 사용자 정의 파일에서 각 요소의 크기를 바이트 수로 표시하며, 10진수로 표시한다. 예 R = 20

· 아래에는 디스크 사용 상태가 표시된다.

BLOCKS FREE … 미사용 블럭 수

BLOCKS USED … 사용된 블럭 수

TOTAL BLOCKS … 디스크의 총블럭 수

⑥ CHAIN

DOS 3.3에서는 기계어 파일을 이용했지만 ProDOS에서는 DOS 명령어로 내장되었다. 이 명령은 BASIC 프로그램에서 사용되며, 지정한 프로그램을 로드시켜 실행시키지만 현재 메모리 내에 있는 모든 변수의 이름과 값을 그대로 보존한다. 따라서 이전의 프로그램이 실행된 결과를 가지고 다시 작업할 수가 있으며 다음에 연결될 프로그램을 위하여 데이타를 보존시킬 수 있다.

CHAIN pathname(, @n) (, Sn) (, Dn)

@n

프로그램의 행번호로 이 행번호에서 시작된다.

⑦ CLOSE

CLOSE(pathname)

지정한 파일을 닫는다. 이때는 버퍼 안의 출력 부분에 있는 모든 문자가 파일에 기록되며, 버퍼는 다른 파일이 쓸 수 있게 된다. 옵션이 없으면 OPEN된 파일을 모두 닫는다.

예 CLOSE

CLOSE/PRODOSBASIC/RECORD

＊ 참고 : 옵션이란 것은 명령어 형식에서 Pathname, An, Txxx, Dn 등을 말한다.

⑧ CREATE

CREATE pathname(, Txxx) (, Sn) (, Dn)

지정된 이름과 형태의 파일을 만든다. 주로 새로운 디렉토리 파일을 만드는 데 유용하다. 예를 들면

CREATE/CHOIYOUNGICK(, TDIR)

위 명령은 /CHOIYOUNGICK이라는 패스네임을 가진 새로운 디렉토리 파일을 만든다.

⑨ DELETE

DELETE pathname(, Sn) (, Dn)

지정한 파일을 디렉토리에서 삭제한다. OPEN되거나 Lock되어 있지 않아야 하고, 디렉토리 파일이라면 비어 있어야 한다. 볼륨 디렉토리 파일은 삭제하지 못한다.

⑩ EXEC

EXEC pathname(, Fn or Rn) (, Sn) (, Dn)

시퀀셜 텍스트(sequential text) 파일의 내용을 직접 모드에서 실행한 것과 같이 차례대로 수행된다. 일단 파일이 OPEN되고 Fn이나 Rn으로 지정한 수의 필드만큼 건너뛰어서 명령을 읽기 시작한다. 파일이 끝나면 닫는다.

한 번에 하나의 EXEC만 동작하기 때문에, 파일 내에 또다시 EXEC 명령이 있으면 원래의 파일은 닫히고 새 파일이 OPEN, 실행된다. CLOSE 명령이 파일 내에 있어도 자신은 닫지 않는다. RUN 명령이 들어있으면 프로그램의 실행이 끝난 후 다음 명령을 실행한다.

예 EXEC XLISTER, F1, S6, D1

⑪ FLUSH

FLUSH(pathname)

현재 파일 버퍼 안에 있는 출력 부분 내용을 디스크에 출력한다. 디렉토리도 이에 맞추어 수정되며 작업 중의 백업 처리와 같다. 옵션이 없으면 OPEN된 모든 파일을 대상으로 한다.

예 FLUSH/TEXT

⑫ FRE

이전의 프로그램들이 사용했던 데이타들을 메모리에서 지운다. 프로그램 안에서 CTRL-D 다음에 쓰면 ProDOS 명령어가 되고 그냥 쓰면 애플소프트 명령어가 된다. 하지만 ProDOS의 FRE가 훨씬 더 빠르고 효율적이다. 간단히 말해서 가베지 컬렉션에 사용한다.

예 PRINT CHR$(4) : "FRE"

⑬ IN#

PR#과 함께 메모리 내의 기계어 루틴을 문자 입출력 루틴으로 쓸 수 있게 해주는 기능이 추가되었다.

IN#n or IN#An

가. IN#n : n은 슬롯의 번호로 해당 슬롯에 접속된 카드상의 롬의 첫바이트($Cn00)가 입력 루틴의 어드레스로 세트된다. 예 IN#6

나. IN#An : n으로 지정한 어드레스에 있는 입력 루틴을 이용하게 된다. 이 루틴도 첫바이트는 반드시 6502 어셈블리 명령 CLD(CLear Decimal)이어야 한다. 예 IN#A$300

⑭ LOAD

LOAD pathname(, Sn) (, Dn)

이 명령이 내려지면 EXEC 파일을 제외한 모든 파일을 닫으며 기존의 애플소프트 프로그램은 지워

지고 지정한 애플소프트 프로그램이 로드된다.

예 LOAD STARTUP

⑮ LOCK

LOCK pathname(, Sn) (, Dn)

지정된 파일을 Lock시켜서 삭제, 변경하지 못하게 한다. Lock된 파일은 *로 표시된다.

예 LOCK/ARCADE/LODE. RUNNER

⑯ POSITION

ProDOS는 READ, WRITE 자체에 이 기능이 있으므로 필요가 없으나 DOS 3.3과의 호환성을 위해 없애지 않았다.

POSITION pathname, Fn or Rn

지정된 숫자의 필드만큼 현재 위치를 앞으로 이동시킨다. Fn과 Rn은 같은 의미로 사용된다.

예 POSITION TEXT, F8

⑰ PREFIX

PREFIX (pathname) (, Sn) (, Dn)

Prefix를 지정하는 데 사용한다. 아무런 옵션없이 사용하면 현재의 Prefix를 알려주며, 프로그램 내에서는 다음의 INPUT 명령에서 입력받는 내용이 Prefix로 된다. Pathname 대신 /를 쓰면 Prefix를 비운다. 이때는 현재 지정된 슬롯과 드라이브의 볼륨 이름만 남게 된다.

예1 PREFIX

예2
```
10 ? CHR$(4) : "PREFIX"
20 INPUT A$
30 ? A$
```

예3 PREFIX/

⑱ PR#

IN#과 같은 기능이 추가되었다.

PR#n or PR#An or PR#n, An

가. PR#n : n은 슬롯 번호로 해당 슬롯에 접속된 카드상의 ROM의 첫 바이트가 출력 루틴의 어드레스로 세트된다.

예 PR#1

나. PR #An : n으로 지정한 어드레스에 있는 출력 루틴을 이용하게 된다. 단, 첫 바이트는 반드시 6502 어셈블리 명령 CLD이어야 한다.

예 PR#A$300

다. PR#n, An : 슬롯 번호와 An이 함께 사용되어서 지정된 어드레스가 해당 슬롯으로의 출력 어드레스를 부여한다. 출력을 전환시켜 주지 않으므로, 이어서 반드시 PR#n을 해주어야 한다.

예 PR#1, A$300…$300번지에 있는 루틴을 사용하여 1번 슬롯으로 출력하는 것이다. 이 방법을 응용하면 다른 카드의 롬에 있는 루틴을 이용하여 출력시키는 것이 가능해진다. 즉 1번 슬롯으로 출력시키되 출력 루틴은 2번 슬롯에 접속된 카드의 루틴을 이용하려면 PR#1, A$C200이라고 하면 된다.

⑲ RENAME

RENAME pathname 1, pathname 2(, Sn) (, Dn)

pathname 1로 지정한 파일 이름을 pathname 2로 바꾼다. 단, 같은 디렉토리 안에서만 가능하다.

예 RENAME TINE, TIME

⑳ RESTORE

RESTORE pathname(, Sn) (, Dn)

현재의 BASIC 변수들을 클리어시키고 지정한 변수 파일(VAR 형태)에 저장한 변수들을 변수 영역에 넣어준다.

예 RESTORE C. Y. I. VAR

㉑ RUN

RUN pathname(, @n) (, Sn) (, Dn)

지정된 애플소프트 프로그램을 로드시켜 실행시킨다.

CHAIN에서와 같이 @n은 실행시킬 문번호이다. 즉 프로그램을 로드하여 @n에서 지정한 문번호에서부터 실행시키고, 해당 문번호가 없을 때에는 다음 문번호에서부터 실행된다.

예 RUN/CHOIYOUNGICK/STORE. RESTORE(, @ 30)

㉒ SAVE

SAVE pathname(, Sn) (, Dn)

현재 메모리에 있는 애플소프트 프로그램을 지정한 이름의 파일로 세이브시킨다. 같은 이름의 파일이 있으면 지워버리고 대체되어 들어간다. 단, 같은 이름의 기존 파일이 Lock되어 있으면 에러가 난다.

㉓ STORE & RESTORE

간단히 말해서 변수 파일에 저장, 재생하는 것이다. 즉 STORE는 저장, RESTORE는 재생이다.

STORE pathname(, Sn) (, Dn)

현재 정의되어 있는 모든 BASIC 변수들을 모아서 지정한 이름의 파일로 디스크에 세이브시킨다 (파일의 종류는 VAR).

세이브하기 전에 애플소프트의 스트링 영역을 압축시키므로 2~4초 정도 지연된다.

RESTORE pathname(, Sn) (, Dn)

현재의 BASIC 변수들을 클리어시키고 지정한 이름의 변수 파일(VAR type)에 저장한 변수들을 변수 영역에 넣어준다.

㉔ UNLOCK

UNLOCK pathname(, Sn) (, Dn)

Lock된 파일을 해제한다.

㉕ WRITE

지정한 파일에서 현재 위치를 이동시켜 내용을 쓴다. 이 명령이 내려지면 이후의 모든 출력 문자는 파일에 써넣어진다. 다른 ProDOS 명령이 내려지면 기능은 끝난다. 디렉토리 파일은 열고 읽을 수는 있으나 쓸 수는 없다.

WRITE pathname(, Rn) (, Fn) (, Bn)

Rn이 사용되면 현재 위치가 지정된 레코드의 첫머리로 이동하며, Fn이나 Bn이 사용되면 지정한 숫자의 필드나 바이트만큼 앞으로 건너뛴다.

㉖ OPEN

OPEN pathname(, Ln) (, Txxx) (, Sn) (, Dn)

지정한 파일에 메모리 버퍼를 할당하고, 파일의 처음부터 읽거나 쓸 수 있도록 시스템을 준비시킨다. 파일이 없으면 새로 만들어진다. Ln이 지정되어 있지 않으면 OPEN될 때의 레코드 길이가 디폴트로 설정되고, 새로운 파일에서는 1로 된다.

Txxx를 사용하면 텍스트가 아닌 파일을 읽거나 쓸 수 있도록 OPEN시킬 수 있다. 이때는 반드시 파일이 존재해야 한다. 이 기능을 이용할 때에는 파일의 내용 취급에 상당한 주의를 해야 할 것이다.

동시에 여덟 개의 파일을 OPEN할 수 있으며, CAT, CATALOG, EXEC, – 명령도 파일을 OPEN시키지만, OPEN 명령만이 파일을 OPEN한 채로 둔다(OPEN된 파일은 반드시 닫아두어야 한다).

예 OPEN SESAME/SEED… 시퀀스 파일을 OPEN한다.

OPEN/×/RECORD, L100… 레코드 길이가 100인 랜덤억세스 파일을 OPEN한다.

OPEN ARCADE, TDIR… 디렉토리 파일을 OPEN한다.

㉗READ

이 명령이 내려지면 다음의 INPUT이나 GET 명령은 파일로부터 내용을 읽어들이며, 캐리지 리턴(ASCII 코드 13, carriage return)이나 224바이트에서 끝난다. 다른 ProDOS 명령이 내려지거나 널(null) 명령이 내려지면 READ 기능은 끝난다.

READ pathname(, Rn) (, Fn) (, Bn)

지정된 파일에서 현재 위치를 이동시켜 내용을 읽는다. Rn이 사용되면 지정된 레코드의 첫부분으로 이동하며, Fn이나 Bn이 사용되면 지정한 숫자의 필드나 바이트만큼 건너뛴다.

2. 에러 메시지(error message)

모두 19개의 에러 메시지가 있는데 DOS 3.3과 같이 에러 메시지 앞에 ?가 없어 애플소프트의 에러와 구별할 수 있다. 시스템 모니터상에서 에러가 발생하면 BASIC으로 리셋되고 메시지가 나온다.

RANGE ERROR(code 2)

옵션의 값이 너무 작거나 클 때 발생한다. 그러나 0~65535를 벗어난 경우에는 Syntax error가 된다.

NO DEVICE CONNECTED(code 3)

카드가 없는 슬롯을 지정(PR#이나 IN# 등)하거나 주변기기가 카드에 접속되지 않은 경우, 드라이브에 디스크가 없는 경우에 발생한다(일부의 드라이브) 카드의 접속 상태도 점검해 볼 필요가 있다.

WRITE PROTECTED(code 4)

디스크에 기록하는 명령이 내려졌을 때 디스크에 라이트 프로텍터가 붙어있으면 이 에러가 발생한다.

END OF DATA(code 5)

텍스트 파일에서 전혀 정보가 수록된 적이 없는 부분에서 읽으려고 하는 경우에 발생한다.

PATH NOT FOUND(code 6 또는 7)

DOS 3.3의 FILE NOT FOUND와 같은 기능으로 지정한 파일이 디스크에 없거나, 패스네임의 규칙에 어긋날 때 발생한다. Prefix가 맞지 않거나 철자가 틀린 경우도 해당된다.

I / O ERROR(code 8)

데이타나 프로그램의 읽기와 쓰기가 실패한 경우 발생한다(ProDOS는 96번의 시도를 해 본 후에 에러로 처리한다). 발생하는 경우는 다음과 같다.

· 드라이브의 도어(door)가 열려 있다.
· 드라이브에 디스크가 없다.
· 디스크가 포매팅이 안되어 있다.
· 디스크의 드라이브내 위치가 잘못되어 있다.
· Dn으로 지정한 드라이브가 없다.
· 디스크가 구형의 13섹터 디스크이다.
· Sn으로 지정한 슬롯에 디스크 컨트롤러 카드가 없다.
· PR#, IN#으로 지정한 슬롯에 카드가 없다(슬롯, 드라이브는 에러가 발생해도 계속 유지되므로 바로잡아 주어야 한다).

DISK FULL(code 9)

디스크가 꽉 차서 더이상 들어가지 못할 때 발생한다. 에러가 나면, 넣으려 했던 것을 디스크에서 삭제하는 것이 좋다.

FILE LOCKED(code 10)

Lock된 파일에 기록, 수정, 삭제 명령을 했을 때 발생한다.

INVALID OPTION(code 11)

존재하지 않거나 부적절한 옵션을 사용했을 때 발생한다.

NO BUFFERS AVAILABLE(code 12)

이미 여덟 개의 파일이 OPEN되어 있는 상태에서 파일을 OPEN시키는 APPEND, CAT, CATALOG, EXEC, OPEN 등의 명령을 사용했을 때 발생한다.

FILE TYPE MISMATCH(code 13)

명령과 맞지 않는 형태를 가진 파일을 지정했을 때 발생한다.

PROGRAM TOO LARGE(code 14)

메모리가 부족하여 파일을 완전히 로드시킬 수 없을 때 발생한다.

NOT DIRECT COMMAND(code 15)

직접 모드에서 쓸 수 없는 APPEND, OPEN, POSITION, READ, WRITE 명령을 직접 모드에서 사용했을 때 발생한다.

SYNTAX ERROR(code 16)

명령어에 구문상의 에러가 있을 때, 패스네임에 허용되지 않는 문자가 들어있을 때, 옵션 문자가 틀렸거나 필수 옵션을 빠뜨렸을 때, 구분 기호(대개,)가 틀리거나 빠졌을 때 발생한다.

DIRECTORY FULL(code 17)

한 디렉토리 파일에는 51개의 파일을 넣을 수 있다. 그러나 51개가 넘으면 이 에러가 발생한다. 이때는 서브디렉토리를 이용하도록 한다.

FILE NOT OPEN(code 18)

OPEN없이 APPEND, READ, WRITE를 실행했을 때 발생한다.

DUPLICATE FILENAME(code 19)

CREATE나 RENAME할 패스네임이 이미 존재할 때 발생한다.

FILE BUSY(code 20)

파일이 OPEN되어 있을 때 CAT(ALOG), DELETE, RENAME을 하면 발생한다. 먼저 닫아주어야 한다.

FILE(S) STILL OPEN(code 21)

한 개 이상의 파일이 OPEN되어 있는 상태에서 프로그램의 실행이 정지됐을 때(다른 에러가 발생하거나 CTRL-C에 의해 인터럽트됐을 때) 발생한다. 다른 LOAD나 RUN 명령 이전에 모든 파일을 닫아주어야만 한다.

사용자에게 매우 편리한 기능의 명령어를 설명하겠다. 바로 대쉬(dash)라는 이름의 명령으로 파일의 형태에 관계없이 실행시킨다. 시스템 파일(SYS type)을 실행시키는 명령은 없는데 이 -(dash)는 실행시킬 수 있다. 단 실행시키면 메모리가 완전히 지워진다. 형식은

- pathname(, Sn) (, Dn)

예 - PRODOS
   - START

이렇게 해서 ProDOS의 에러 메시지들까지도 살펴보았다. 메시지 앞의 코드 번호는 에러 코드로서 프로그램상에서 ONERR GOTO 문으로 에러를 제어할 수 있도록 적어놓은 것이다. 아래 예를 보면서 설명하겠다.

예
```
10  ONERR GOTO 100
20  D$= CHR$(4)
30  PRINT D$, : "CATALOGE"
40  END
100  EC= PEEK(222)
110  IF EC= 16 THEN PRINT "SYNTAX ERROR" : END
120  PRINT "ERR" : END
```

위 프로그램을 살펴보면 30행에서 명령어에 이상이 있는 것을 알 수 있다. 위 프로그램을 실행시키면 10행에 의해서 30행에서 100행으로 가게 되고 100행에서는 PEEK(222)로 어떤 에러인가를 에러

코드로서 체크하고 30행에서 SYNTAX ERROR가 있으므로 100행의 EC변수에는 16이 저장되고 110행에 의해 멈추게 된다.

〈참고 자료〉

ProDOS Technical Reference Manual
BASIC Programming With ProDOS
COPY Ⅱ PLUS 6.3 Manual by Central Point Software
컴퓨터 학습 87년 6 · 8월호

제 8 장

애플 IIe의 통합 소프트웨어 AppleWorks

§ 1. 개요

AppleWorks는 애플사에서 만든 최고의 사무용 편집 통합 소프트웨어이다. 이 프로그램은 사무용 유틸리티로서 초보자가 사용하기에는 약간의 난이한 점(한글 문제)들이 있으나 AppleWorks V2.0이 100% 활용되기만 한다면 현재 시중에 나온 16비트 컴퓨터에 뒤지지 않을 만큼의 능력을 발휘할 수 있을 것이다. 또한 속도면에서도 다른 어떤 유틸리티에 뒤지지 않을 만큼의 빠른 편집 기능과 전송 기능을 보유하고 있고, 데이타의 호환성 문제도 많이 해결되었다.

원래 이 프로그램은 현재 시중에 나온 애플Ⅱe, Ⅱc 전용 유틸리티였는데 몇년 전 Ⅱ+에서도 가동될 수 있게 하는 Ⅱ+용 AppleWorks가 나왔다. 그러나 그 시스템에서의 능력은 하드웨어의 제약으로 인해 불완전해지게 되어 이를 보완, 절충하여 완전한 Ⅱe용 S/W로 만든 것이 바로 AppleWorks V2.0이다.

(1) 갖추어야 할 시스템

- 애플Ⅱe(기본 메모리 64K+보조 메모리 64K), Ⅱc 호환 기종
- 디스크 드라이브 : 한 대 이상
- 프린터 : 한 대
- AppleWorks V2.0 프로그램 디스켓 세 장

1장 : Start up : AppleWorks의 Start 프로그램 ┐
Program : 메인 프로그램 ┘ 메인 디스켓

2장 : Samples Data ┐
Samples Data(volume name : intermediate) ┘ 데이타(예제) 디스켓

3장 : "Presents"라고 해서 AppleWorks V2.0을 사용하기에 기본적인 지식과 사용법을 연습과 퀴즈로 알기 쉽게 풀어주고 있다.

- 하드디스크 드라이브(ProFile) : AppleWorks ProDOS를 사용하므로 트리 구조의 파일을 다량 관리할 수 있는 ProFile을 사용한다.

(2) AppleWorks V2.0의 특징

① 완벽한 편집 기능

Review/Add/Change 모드에서의 COPY 기능과 MOVE 기능, Cut & Paste 기능들을 보유하고 있으며 또한 한 음절의 글자부터 문장에 이르기까지 데이타에 수록되어 있는 것이라면 그 어떠한 문자라도 찾을 수 있는 Find 기능을 함께 보유하고 있다.

② 강력한 처리 속도 기능

AppleWorks는 새로운 시스템 형태인 ProDOS를 사용하기 때문에 그 처리 속도는 16비트용 유틸리티 못지 않게 빠르다. 예를 들면 로딩, 세이빙의 속도와 새로운 방법의 포매팅 처리와 Edit 모드에서의 Ruler 기능은 애플 컴퓨터에서의 혁신이라 할 수 있다.

③ 안내 기능

AppleWorks는 언제 어느 곳에서든지 자체의 Guide 기능을 보유하고 있다. 각각의 프로그램마다 쓰일 수 있는 기능들이 제각기 다르기 때문에 독자들은 필히 사용해보기 바란다.

④ 데이타 호환 기능

AppleWorks의 구조를 살펴보면 크게 워드프로세서, 데이타베이스, 스프레드쉬트의 세 가지로 이루어져 있다. 이들은 데이타를 공유할 수 있는 것은 물론, 다른 S/W의 데이타를 이용할 수 있는 호환성이 있다.

§ 2. AppleWorks의 기본적인 이해

1. AppleWorks의 구성

AppleWorks는 세 가지의 기능을 통합한 프로그램이다. 이는 데이타베이스, 워드프로세서, 스프레드쉬트와 같이 세 개의 응용 소프트웨어에 의해 정보를 생성한다.

각 소프트웨어의 응용 부분을 간단히 요약해 보면 다음과 같다.

AppleWorks
- 데이타베이스 : 재고관리, 시간계획, 거래처 명부, 고객관리 등의 자료들을 갱신, 추가, 삭제, 조회하는 작업이 가능하다.
- 워드프로세서 : 보고서, 메모, 편지, 도표작성 등의 작업이 가능하다.
- 스프레드쉬트 : 예산, 재정, 가계부, 지출명세서, 통계학적 계산 등의 작업이 가능하다.

위에서와 같이 세 가지의 독립된 입력 데이타를 "Clipboard"라는 임시 기억 장치에서 자르고 붙이는 (cut & paste) 작업을 할 수 있다.

예를 들면, 종전에는 스프레드쉬트에서 처리한 데이타를 보고서에 같이 출력하려면 각 소프트웨어에서 따로 출력하여 직접 자르고 붙이는 불편함이 있었다. 그러나 이 AppleWorks에서는 보고할 워드프로세서 데이타에 스프레드쉬트 데이타를 원하는 곳에 자르고 붙이는 작업을 화면상에서 키조작으로 간단하게 처리한다.

그림8.1은 AppleWorks의 구성을 나타낸 그림이다.

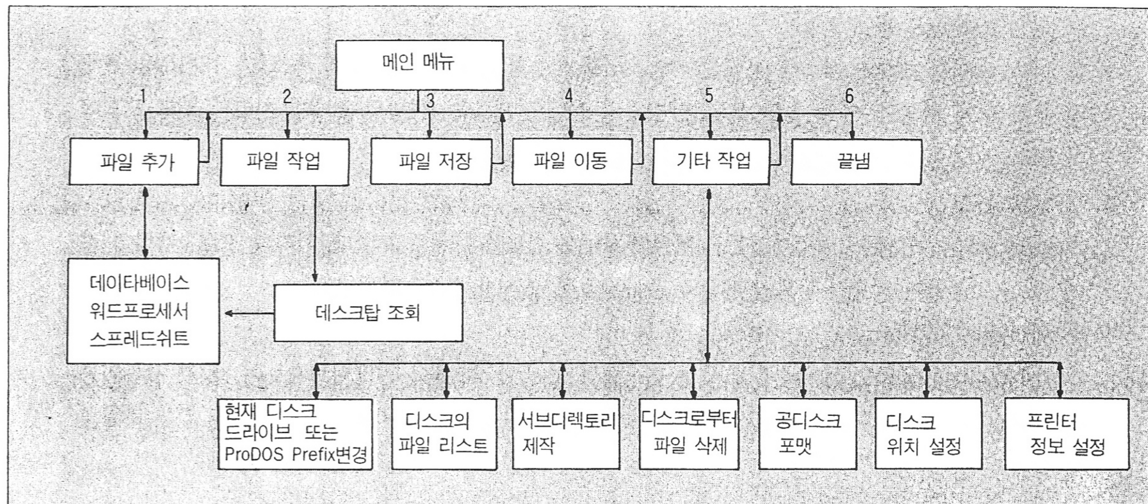


그림 8.1 AppleWorks의 구성

2. AppleWorks의 파일

AppleWorks는 세 가지 소프트웨어의 데이타(데이타베이스, 워드프로세서, 스프레드쉬트)를 상호 이용하는 것이 가능할 뿐만 아니라, Visicalc나 DIF, Quick File, ASCII 파일의 데이타 처리도 가능하다.

AppleWorks File ┌ 워드프로세서 : ASCII File
├ 데이타베이스 : ASCII File, Quick File, DIF File
└ 스프레드쉬트 : DIF File, Visicalc File

§ 3. AppleWorks V2.0의 사용 방법

1. 메인 메뉴의 사용 방법

(1) Add files to the desktop

① 데이타 디스켓에서 파일을 읽어들여 데스크 탑에 추가시키는 기능을 수행한다.

· The current disk : 접두어명 또는 드라이브 번호를 찾아서 디스크 안의 파일 리스트가 알파벳 순서로 출력된다.

· A different disk : 위에서 지정된 접두어나 드라이브 번호를 바꾸고 싶을 때 사용한다.

· 워드프로세서 : 전문적으로 문서 작성.
· 데이타베이스 : 자료의 유지 및 효율적 관리.
· 스프레드쉬트 : 수치 계산이나 함수 계산.

②만약 메모리 공간에 이미 입력되어 있는 파일을 또 다시 추가 로딩하려고 한다면 확인을 요한다.

┌ Yes : 파일에 관계없이 추가로 재입력된다.
└ No : 재로딩은 성립되지 않고 데스크탑의 파일들은 그대로 보전된다.

③데스크탑의 메모리 용량은 전부 12개의 파일을 입력시킬 수 있다. 로딩이 되면 곧바로 Edit 모드로 들어가게 된다.

(2) Work with one of the files on the desktop

데스크탑에 있는 파일들 중에서 하나를 뽑아내어 작업(edit)하는 것을 뜻한다. 선택하면 데스크탑의 리스트가 나오고 목록에는 어떤 소프트웨어에서 에디트된 것인가를 알려준다.

예 Rolodex DB : 데이타베이스에서 만든 Rolodex라는 이름의 파일을 뜻한다. 파일을 선택하면 Edit 모드(Review/Add/Change)로 들어간다. 언제든지 'ɞ-Q'를 입력해 주면 데스크탑의 파일 리스트가 화면에 나타난다.

(3) Save desktop files to disk

지금 사용하고 있는 데스크탑의 파일들을 디스크에 세이브시킨다. SAVE FILES가 나오면 ↓, 커서키를 이용하여 선택한 후 리턴키를 누른다. 그리고 잠시 후에 Current Disk (지정된 디스크)에 세이브할 것인가 아니면 다른 디스크 드라이브나 하드디스크 드라이브에 세이브할 것인가를 묻는다.

이때에 Current Disk를 선택하면 지정된 디스크에 세이브가 이루어진다. 만약 세이브하려는 파일명이 이미 데이타 디스켓에 수록되어 있다면 기존의 파일을 제거하고 새로운 데이타를 입력할 것인지 아니면 파일명을 바꾸어서 세이브할 것인지를 묻게 된다. 알맞는 파일명을 입력해 주고 리턴키를 누르면 된다.

• First change a different disk or directory : 디스크 디렉토리나 지정된 디스크의 번호를 바꿀 때 사용하는 기능이다.

(4) Remove files from the desktop

메모리되어 있는 데스크탑의 파일을 삭제하는 기능이다. 선택하면 화면 중간부에는 출력되는 파일의 상태에 따라서 그 Delete의 진행 방법이 제각기 틀리다.

• New : 새로 만든 파일(현재 Edit 과정의 파일) ┐ ① 상태
• Changed : 기존의 파일을 약간 수정한 것 ┘

- Unchanged : 아무런 처리를 안한 순수한 파일
- Saved : New 파일이라 해도 이미 데스크에 수록되어 있는 파일 ② 상태

①번 상태의 파일에서의 삭제 방법은 세 가지가 있다. 첫째 데스크탑에서 삭제하되 지정되어 있는 디스크에 세이브하고 삭제한다. 둘째 다른 디스크나 디렉토리로 변경시킨 후에 세이브시킨 다음 삭제한다. 세째 무조건 파일을 삭제만 시킨다.(Yes/No)

②번 상태 : 삭제한다.

(5) Other Activities

AppleWorks를 다루면서 부수적으로 쓰이는 장비들을 골라 모아놓는 곳이다. 선택하면 서브메뉴가 나온다.

① Change current disk drive or Pro.DOS prefix

지정 디스크나 ProDOS의 접두어를 변경시키는 기능이다. Change current disk 메뉴가 나오면 원하는 것을 선택, 입력시킨다. 하드디스크나 다른 ProDOS의 접두어를 선택했을 때에는 접두어를 입력해 준다. 한번 입력해 준 접두어나 디스크 번호는 계속 유효하다.

② List all files on the current disk drive

지정된 디스크의 파일 리스트가 출력되는 기능이다. 서브디렉토리를 사용한 디스켓일 경우에는 다른 기능에서 리스트하고 싶은 그룹명을 미리 입력시켜 놓아야 한다. 접두어를 사용한 디스켓은 그 접두어 그룹의 할당 범위에 있는 파일만을 리스트해 준다.

③ Create a subdirectory

새로운 서브디렉토리를 만드는 기능으로서, Current Disk나 하드디스크에새로운 이름의 서브디렉토리를 만들어서 그 디스켓 속의 파일들을 조직, 그룹화시키는 기능이다. 사용 방법은 이 기능을 선택한 후 서브디렉토리의 완전한 Pathname을 타이프해 주고 리턴키를 누른다.

④ Delete files from disk

이 기능은 데스크탑의 메모리에서 파일을 삭제시키는 것이 아니라 직접 디스크에 들어가 디스켓 속의 파일을 삭제시키는 기능을 가지고 있다. 사용 방법은 커서키를 이용, 선택하고 리턴키를 누르면 삭제된다.

⑤ Format a blank disk

공디스켓이나 다른 어떤 디스켓을 ProDOS로 초기화하고 싶을 경우에 이 기능을 선택해 주고 볼륨명을 입력해 준다. 볼륨명은 AppleWorks에서 상당히 중요한 부분을 차지하고 있으므로 정확하게 입력해야 한다.

⑥ Change standard location of data disk

데이타 디스크의 Standard location을 설정하는 기능으로서, AppleWorks가 처음 시작할 때 일일이

접두어를 타이프해 주지 않아도 자동적으로 데이타 디스켓의 접두어가 미리 왼쪽 상단에 표시해 주는 기능을 담당하고 있다.

⑦ Specify information about your printer(s)

프린터를 설정하는 기능으로 AppleWorks를 100% 활용하려면 필히 거쳐야만 될 관문이라 하겠다. 다음의 프린트 항목에서 설명키로 하겠다.

§ 4. 데이타베이스의 사용 방법

1. 데이타베이스의 구조

AppleWorks의 데이타베이스는 크게 두 가지의 주요 기능을 수행하는데, 이에는 보고서 작성과 정보의 검색, 갱신, 변경의 기능이 있다.

검색, 갱신, 변경 기능을 사용할 때에는 새로운 정보의 갱신, 추가, 삭제, 조회 등 정보를 조직화하여 현상태로 유지시키며 필요시에는 특정 정보만 출력할 수 있다.

보고서 작성 기능은 보고서의 작성과 출력의 기능을 가지고 있으며 정보의 출력을 사용자가 원하는 형태로 구성하며 또 이들은 데이타베이스 파일의 일부가 될 수 있다.

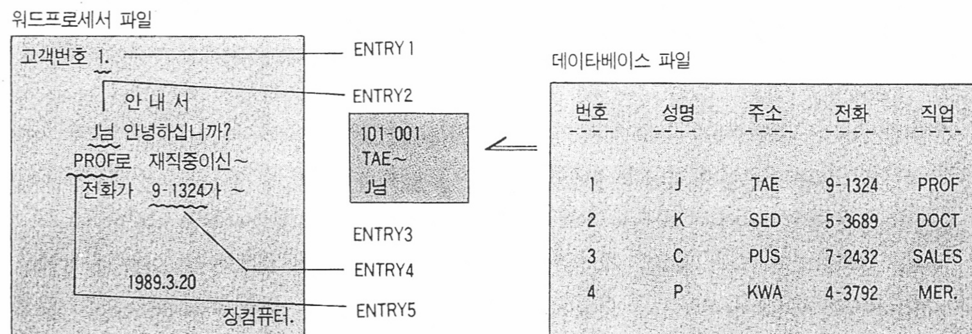


그림8.2 데이타베이스를 이용한 문서 작성

(1) 데이타베이스의 관련 사항

① Data : 정보의 가장 기본이 되는 최소 단위

② Field : 데이타를 구성하는 각 항목

③ Record : 개별 데이타들의 모임

④ File : 레코드가 모여 구성된 전체의 데이타 모임

⑤ Data Base : 총괄적인 데이타의 모임

(2) 데이타베이스의 파일 형태 및 범위

다음은 데이타베이스 파일의 형태와 범위에 대하여 알아보도록 하자. 표 8.1은 AppleWorks의 데이타베이스 범위이다. 기타 dBASE Ⅱ, Ⅲ, Ⅲ+ 등은 각 매뉴얼에서 규정된 범위가 각각 다르다.

파일의 형태	범 위
최대의 Record의 수 (record의 크기나 category의 수와는 무관)	1,350
Record의 크기가 평균 75Characters 일 때 최대 Rocord의 수	64KRAM일 때 약 140 128KRAM일 때 약 750
한 Record당 최대 Category의 수	30
Record의 최대 길이	1,024 Characters
Entry의 최대 길이	76 Characters
한 Category에 들어가는 최대 Character 수	20

표 8.1 데이타베이스 파일의 형태와 범위

AppleWorks에서는 두 가지의 파일 구조를 가지는데 단일 레코드 구조는 Category가 수직으로 배열하여 한 번에 12개의 Entry를 보여준다. 다중 레코드 구조는 Category가 수평으로 배열하여 한 번에 15개의 Record가 리스트된다.

단일 구조와 수직 구조의 전환을 ⍝-Z로 한다.

2. 데이타베이스에서의 편집 방법

편집 순서는 메인 메뉴에서 Add files to the desktop을 선택→ Add 파일 메뉴에서 4번 Data Base 선택→ Data Base 메뉴에서 기능을 선택한다.

① From Scratch : 새로운 파일을 만들 때 사용하며 파일 이름은 데스크탑에 입력한다.

새로운 카테고리의 입력 방법은 맨 처음의 카테고리의 위치에서 CTRL-Y로 초기화시키고 편집키를 사용(↑ : 전 파일로의 이동이나 파일명을 고칠 때 사용한다. ↓ : 다음 파일로의 이동, ⍝-I로 새로운 카테고리를 입력시킨다. Review/Add/Change 모드에서는 이 기능이 레코드의 재삽입시에 사용된다. ⍝-D는 카테고리를 삭제한다)하여 입력시킨다.

㈜ Review/Add/Change 모드에서의 카테고리의 입력은 "⍝-N"이다.

② From a Text(ASCII) : Pascal File이나 APPLE WRITER File들을 AppleWorks의 파일 소스로 사용할 수 있고 데이타베이스에 입력시킬 수도 있다. 그러나 애플의 소스로 활용하기 위해서는 조건에 맞는 파일이어야 한다.

· 각각의 엔트리는 줄(line)로 인해 구분되어야 하며 리턴키로 구분해야 한다.

· 엔트리와 카테고리의 순서는 전파일과 동일해야 한다.

· ProDOS로 입력되어 있어야 하므로 볼륨명을 가지고 있어야 한다. 선택되면 Pathname을 입력해 주고 카테고리의 숫자를 입력해 주어야 한다.

③ From a DIF File : Visicalc에서 만든 DIF 파일을 사용할 수 있다. Pathname과 새로운 이름을 입력해 준다.

④ From a Quick File : Quick File 프로그램에서 쓰인 파일을 사용한다.

3. 데이타베이스에서 이용 가능한 명령어

① ♂ - A : 데이타를 재배열시키는 기능으로서 선택하면 다음과 같은 메뉴가 나온다.

From A To Z : 알파벳 순서로 배열
From Z To A : 역순으로 재배열
From 0 To 9 : 숫자(날짜, 수치, 값)로 배열
From 9 To 0 : 역순으로 재배열

② ♂ - C : 일정 파일을 선택하여 그 파일을 다른 장소에 이동 · 전송시키는 기능이다.

· Current Disk : 현재 위치해 있는 레코드에서 최대 99개의 레코드를 아래로 복사한다.

· To Clipboard : 이 기능은 현재의 레코드를 복사해 두어서 나중에 사용할 수 있는 장점을 가지고 있다.

· From Clipboard : 입력되어 있는 레코드를 다른 파일의 문서나 레코드에 복사 · 이동해 주는 기능이다. 만약 Clipboard에 아무런 레코드가 입력되어 있지 않다면 이 기능은 수행되질 못한다.

③ ♂ - D(delete): 레코드의 삭제 기능으로서, 커서키를 사용한다.

④ ♂ - F : Find 기능으로서, 레코드의 이름을 입력해 주면 메모리에서 찾을 수 있다.

⑤ ♂ - I : 새로운 레코드의 재입력시에 사용되는 기능으로서 선택되어지면 단일 레코드로 화면이 바뀐다. 단일 레코드와 다중 레코드의 예제 화면은 그림 8.3, 그림 8.4와 같다.

㈜ 그림 8.3의 왼쪽 상단의 표시는 현재의 레코드가 지금 메모리돼 있는 레코드들 중에 몇번째라는 것을 의미한다.

⑥ ♂ - L : 레코드의 구조를 변경시키는 작업으로서 데이타에는 영향을 미치지 않고 단지 디스플레이 되는 방법만을 변경시키는 기능을 수행한다. 그러나 카테고리의 삽입 또는 삭제시에는 지정해 둔 디스플레이 구조는 변형된다.

· 다중 구조(zoom out)에서의 변환 작업

다중 구조시, 즉 그림 8.4에서 ♂ - L을 누른 후 리턴하면 그림 8.3과 같은 변환 작업 모드로 들어간다. 화면 구성은 위에는 사용 옵션에 대한 설명이고 아래에는 샘플 레코드 세 개가 출력되어 있어 수월하다. 그럼 여기서 다중 레코드 변환 작업의 옵션에 대해 설명하겠다.

→, TAB : 다음의 카테고리로 커서만을 이동시킨다.

```
File User GrOUP            REVIEW/ADD/CHANGE            Escape: Main Menu

Selection: All records

Record 1 of 30
============================================================================
Name: Jang Y.D

Tel. No.: (053)000-000

St.Address: 199-9 H-M

City: Daegu

Zip: 720-030

Communication: Yes

Com.Parameter: 8bit.none
      .
      .
      .
      .
      .
      .
      .
----------------------------------------------------------------------------
Type entry or use A commands                                   A-? for Help
```

그림 8.3 단일 레코드 예제 화면

```
File: User Group            REVIEW/ADD/CHANGE                 Escape: Main MENU

Selection: All records

Name      Tel. No.     St.Address     City   Zip     Communication    Com.Parameter
=================================================================================
Jang Y.D (053)000-000 199-9 H-M   Daegu  720-030      Yes             8bit.none
Lee K.W  (000)000-000 1423 J-K    Seoul  120-000      Yes             8bit.none
Han S.S  (000)000-000 4427 Y-B    KK     220-109      Yes             8bit.none
Kim Y.M  (000)000-000 210 S-P     Busan  450-230      Yes             8bit.none
Jung Y.S (000)000-000 320 J-P     KJ     780-230      Yes             8bit.none
Jo I.D   (000)000-000 1443 L-p    DH     460-340      Yes             8bit.none
Kim T.I  (000)000-000 128 P-A     JU     210-488      No              No
Park I.M (000)000-000 4768 I-0    IO     320-346      No              No
Go J.Y   (000)000-000 3467 Y-K    DF     355-376      Yes             8bit.none
Kim H.J  (000)000-000 3521 NY     HT     230-287      Yes             8bit.n
  .            .          .         .        .          .               .
  .            .          .         .        .          .               .
---------------------------------------------------------------------------


Type entry or use A commands                                  A-? for Help
```

그림 8.4 다중 레코드 예제 화면

←, ♂ - TAB : 역순으로 커서 이동.

♂ - 〉: 현재 커서가 위치한 카테고리를 오른쪽 카테고리와 비교, 대체해 준다.

♂ - 〈 : 역순으로 커서가 위치한 지점에서 왼쪽과 비교, 교체시켜 준다.

♂ - → : 커서가 위치한 카테고리의 컬럼의 길이를 오른쪽으로 늘려준다. 최대 확대 범위는 80컬럼 화면에서 오른쪽 끝까지이다.

♂ - ← : 카테고리의 길이를 왼쪽으로 축소시킨다.

♂ - D : 카테고리의 삭제시에 사용한다.

♂ - I : 전에 지워졌던 레코드들을 재입력시켜 준다.

ESC키를 사용, Review/Add/Change 모드로 돌아온다.

주 이때의 편집은 화면에 출력되어 있는 샘플 레코드에만 국한되는 것이 아니라 모든 레코드에 영향을 미치는 것이다.

· 단일 레코드에서의 변환 작업(zoom-in)

화면상의 편리나 편집성의 수월함을 원하는 독자들은 단일 구조에서의 변환을 선택하는 것이 쉬울 것이다.

단일 레코드에서의 변환 작업시 기능을 살펴보면,

♂-Z키를 사용, 단일 구조로 들어간다. ♂-L을 입력하여 단일 레코드의 변환 모드로 들어간다.

주 단일 레코드의 편집은 단일 레코드에만 국한되는 것이지, 다중 구조에는 아무런 영향을 미치지 못한다. 물론 다중 레코드 역시 마찬가지이다.

←, →, ↑, ↓, or RETURN : 카테고리에서의 커서 이동

♂-←, ♂- →, ♂-↑, ♂-↓ : 카테고리와 거기에 딸린 레코드(엔트리)까지도 한꺼번에 이동시켜 준다. 사용 방법은 커서를 변경하려는 카테고리의 명칭의 첫자에 커서를 위치해 놓고 편집키를 사용, 이동시킨다. 역시 ESC키로 복귀한다.

⑦ ♂ - M : 화면상에서의 Cut & Paste 기능으로서, 단일 구조에서는 쓰이질 못한다. 이 사용 방법도 역시 클립보드를 사용하는 방법으로서 전송하고 싶은 레코드에서 ♂-M을 선택하면 된다. 많은 레코드를 전송하고 싶은 경우에는 커서키를 사용하여 지정하고 그렇지 않을 경우에는 그냥 리턴한다. 전송된 레코드의 사용 방법을 Edit모드에서 언제든지 ♂-M 하고 From Clipboard를 선택하면 커서가 위치한 레코드 밑으로 메모리되었던 레코드가 출력된다.

⑧ ♂- N : 앞에서 설명한 From Scratch의 옵션과 같다.

· 카테고리의 이동 : ↑, ↓

· 카테고리 명칭의 변경 : 변경하려는 카테고리에 커서를 놓고 편집 기능을 이용하여 변경하고 리턴한다.

· 새 카테고리의 삽입 : 원하는 위치에서 커서를 지정하고 ♂-I를 누른다. 현재의 구조를 표준 구조로 되돌릴 것인가에 Yes로 답한다. 새 카테고리를 입력해 준다.

· 카테고리의 삭제 : 커서를 위치해 놓고 ♂-D를 선택한다. 변경이 끝나면 ESC로 복귀한다.

⑨ ♂-P : 데이타베이스를 이용하여 새로운 레코드를 만드는 기능(리포트 기능)이다. 설명은 차후에 하기로 하자.

⑩ ♂-R : 하나의 특정한 레코드를 선택하여 편집하려는 레코드의 중심으로 만든다.

⑪ ♂-V : 레코드의 표준 수치를 리스트해 준다. 편집키를 사용하여 표준 수치를 재입력시킬 수 있다.

⑫ ♂-Z : 다중, 단일 레코드의 변환 스위치.

⑬ ♂-Z : Clipboard를 사용치 않고 COPY하는 기능으로서, 커서 위치에 있는 레코드를 그 자리에서부터 COPY하는 기능이다. 참고로 커서의 위치만을 COPY한다.

⑭ RETURN : 모든 옵션 내에서의 수행 가능이다.

⑮ TAB : 다음 카테고리로 이동한다(오른쪽)

♂-TAB : 반대로 이동시킨다.

♂- ↑ : AppleWorks의 화면은 한 번에 15개의 레코드를 출력시키므로 나머지 부분의 화면을 볼 수 있게 하는 이 기능은 Full Screen으로 전의 화면을 볼 수 있다.

♂- ↓ : 다음의 화면을 볼 수 있다.

♂- 1 Through 9 : Ruler 기능이라고 해서 전체의 레코드를 9등분하여 출력시켜 준다.

♂- S : 선택되면 현재 편집중인 파일은 세이브시킬 수가 있다. 세이브는 현재의 파일 이름으로 입력되며 Disk로는 Current Disk에 입력된다. 세이브를 중단시키려면 ESC키를 누르면 된다. 중단되면 원래의 모드로 복귀한다. 이로써 Help 기능에서 볼 수 있는 List 기능들의 분석을 마친다.

4. 데이타베이스에서의 내용 입력 방법

♂- E : 삽입 커서와 삭제 커서의 전환

Del : 커서 위치에서 왼쪽의 글자를 삭제

ESC : 엔트리의 첫글자로 커서 이동

←, → : 변화없이 커서만의 위치 이동

CTRL- Y : 커서 위치에서부터의 오른쪽 삭제

주 편집시의 커서는 언제나 삽입 커서이다.

↓(커서가 엔트리에 첫자 위치시) : 하나 밑의 엔트리로 이동할 때

↑(커서가 엔트리에 첫자 위치시) : 하나 위의 엔트리로 이동할 때

단일 구조에서 사용하면 매우 쉽게 구할 수 있다.

♂- ↓, ♂-↑ : 전체의 레코드를 리스트해 보고 싶을 때

§ 5. 데이타베이스에서의 리포트 작성 방법

1. 기본적인 사양

리포트의 형식으로는 두 가지의 경우가 있는데 제각기 장점이 있다.

• 도표식 : 소계, 통계시에 사용

• 레이블식 : 카테고리의 위치 선정에 유용

2. 리포트의 작성 방법

작성하고 싶은 해당 파일에서(Review/Add/Change) ♂- P를 누른다. 메뉴에는 여섯 개의 옵션이 있다.

• Get a report format : 전에 사용한 리포트, 형식(레이블, 도표식)을 채택하여 편집하는 기능이다. 형식을 선택하면 해당된 형태로 편집된다.

• Create a new "tables" format : 도표 형식의 리포트를 작성한다.

• Create a new "lables" format : 레이블 형식의 리포트를 작성한다.

• Dupliate a existing format : 기존의 포맷을 사용하여 이름이나 간단한 수정이 필요할 때 쓰이는 기능이다. 사용 방법은 새로운 이름의 형태를 입력해 주고 디스플레이되는 형태를 편집키로 수정하는 기능이다.

독특한 편집키 : ♂- V : 카테고리에 따른 엔트리가 출력된다.

♂- Z : 카테고리가 지워지고 그 위치에 엔트리가 출력된다.

• Erase a format : 포맷을 삭제한다. 확인을 요한다.

• Keep working with current of format : 이 옵션은 복귀 기능으로 최근에 사용했던 포맷 형식으로 이동된다.

(1) 도표식 리포트의 작성 방법

Review/Add/Change 모드에서 Create a new "tables" format을 선택하면 편집 화면으로 들어간다. 화면 구성은 간단한 옵션과 샘플 데이타로 구성되어 있다. 각각의 기능을 살펴보자.

←, → : 카테고리 사이에서의 커서 이동

♂- →, ♂- ← : 컬럼 넓이의 확장, 축소

♂- 〉: 오른쪽 카테고리와의 교체 작업

 - 〈 : 왼쪽 카테고리와의 교체 작업

 - D : 카테고리의 삭제

 - I : 삭제된 카레고리의 재삽입

 - R : 처음 포맷 작성시 미리 정해져 있는 선택 기준을 -R로 이용, 변환시킨다.

 - A : 재배치

 - J : 계산 카테고리를 만들기 위한 기초 작업으로 알파벳 정보일 때는 "①"을 입력, 숫자의 경우 자리수를 입력해 주고 각 엔트리의 뒤에 있는 스페이스바의 수를 입력해 준다(프린팅시 이 기능을 체크한다).

 - K : 계산 카테고리를 만들기 위한 기초 작업으로 수에는 ×, +, -, /를 사용한다. 갯수와 카테고리의 스페이스를 정해준다.

 - T : 카테고리의 총계와 소계를 낼 수 있는 기능으로 카테고리의 밑에서 2중선으로 쳐진 것이 있다면 합계가 있는 것이다. -G를 사용하여 그룹 합계를 지정, 선택할 수 있다.

(2) 레이블식 리포트의 작성 방법

Review/Add/Change 모드에서 "Create a new" lables format을 사용하여 그곳에서 - P를 선택한다.

각각의 기능을 살펴보면 :

←, →, ↑, ↓, RETURN : 커서만을 이동.

- (←, →, ↑, ↓) : 이동하려는 카테고리의 첫자에 커서를 위치시키고 사용.

 - D, -I : Cut & Paste 기능에 못지 않는 기능을 발휘한다.

 - J : 카테고리 리스트중에 현재 출력된 레코드를 오른쪽의 것이 왼쪽으로 이동시키는 기능이다.
이때도 역시 커서를 카테고리 첫자에 위치해야 한다.

 - A : 재배열 기능.

 - R : 앞에서 설명된 바와 같다.

 - Z : 레코드의 내용 점검에 사용.

 - N : 레코드의 이름이나 타이틀을 재입력하고 싶을 때 사용.

(3) 데이타베이스의 출력

편집 기능의 마무리 지점이라고 할 수 있는 이 부분에서는 어떻게 데이타를 프린트할 것인가에 대해서 미리 설정해놓는 프린트 옵션 기능이다. 이 옵션들은 어떻게 프린트할 것인가에 대해서만 영향을 미칠 뿐이지 프린터 기기의 선택 기능과는 다른 옵션들이다. 프린터 기기의 선택이라면 "Other Activities"

에서 7번 "Specify information about your printer(s)"의 옵션이 담당하고 있다.

① 리포트 형식에 따른 프린트 옵션들의 차이

· 도표식 리포트 형식의 프린트 옵션 : 줄간격(SS, DS, TS) 상하 간격, 좌우 간격, 포매팅

· 레이블식 리포트 형식의 프린트 옵션 : 줄간격(OL, KS) 상하 간격, 좌우 간격, 포매팅

SS : Single Space

DS : Double Space

TS : Tripple Space

OL : Omit Line

KS : Keep number of lines the Same within each record

② 사용 방법(공통적으로 쓰이는 기능)

리포트 화면에서 ㆁ- O를 선택 → 변경하려는 해당 코드의 입력을 2문자(예 : PW, PH, RM…)로 타이핑해 주고 알맞은 값을 입력하면 된다.

각각의 기능을 살펴보면,

· 좌우 간격 옵션

PW : 프린터의 용지 폭

디폴트 : 8.0인치, 최대값 : 13.2인치

LM : 종이의 왼쪽 끝과의 Marzine

디폴트 : 0인치, 최대값 : 9.0인치

RM : 페이퍼 오른쪽 끝과의 Marzine

디폴트 : 0인치, 최대값 : 9.0 인치

CI : 한 인치당 프린트되는 문자의 수

디폴트 : 한 인치당 10자

인치에 맞는 프린팅의 값(4~24정도)

레코드와 레코드 사이의 상하 간격 옵션

PL : 사용하는 프린트 페이퍼의 수직 길이를 입력.

디폴트 : 11인치, 최대값 : 25.4인치

TM : 프린트되는 첫문장의 시작 거리

디폴트 : 0, 최대값 : 9인치

㈜ 수동 조절이 가능하다.

BM : 끝 문장의 프린트가 페이지 하단부와의 거리를 입력.

디폴트 : 2인치(도표식) 0인치(레이블식), 최대값 : 9인치

LI : 인치당 줄의 수

디폴트 : 6줄, 사용 범위 : 6~8

· 포매팅 옵션

SC : 이 선택은 프린터가 사용할 수 있는 특수 코드의 프린팅 효과가 있다. 사용 중에 프린터의 능력을 확인해 보아야 한다.

PD : 엔트리에 내용이 없을 경우 Dash(-)를 찍게 하고 카테고리가 정해지면 Zero를 프린트한다. 디폴트는 관계없다.

PH : 리포트의 모든 단서(pathname, 날짜, 이름, 타이틀 이름)를 프린팅한다. 디폴트는 Yes로 되어야 하며 No로 선택되어져 있으면 페이지의 제목만이 프린트된다.

줄간격 옵션

도표식 : 줄간격을 입력하는 기능으로 1줄, 2줄, 3줄의 메뉴만으로 쓸 수 있다. 변경하려는 해당 코드만 타이핑하면 된다.(SS/DS/TS)

레이블식 : OL : 디폴트가 Yes로 그대로 있으면 비어있는 엔트리의 경우에는 프린트를 생략한다. 반대로 No로 되어 있다면 엔트리가 없더라도 Dash(-)가 표시되는 기능이고 KS 옵션도 없어진다.

KS : OL의 옵션에서 Yes이고 이 옵션의 디폴트가 Yes이면 레코드 밑에 빈 줄을 추가함에 따라서 모든 레코드에 같은 줄 수를 가지게 해준다.

OL이 Yes이고, 옵션이 No이면 엔트리가 있을 경우 각 레코드의 줄 수가 달라진다.

§6. 워드프로세서

(1) 워드프로세서의 기능

① Work with information in documents

문서의 작성 · 수정 · 편집, 문서내 일정 범위의 삭제 · 이동 · 복사, 내용의 대체, 내용의 검색 등.

② Format documents

형태 · 배열의 지정, 간격의 조정, 페이지 부여, 특수한 프린팅 방법의 사용.

예 Boldface, Underline 등.

(2) 파일의 사양

① 파일의 최대 길이

줄간격 한 줄, 페이지당 54줄이면 28페이지에 해당한다.

② 최대 문자 수

10,000자(64KRAM) 약 8페이지

56,000자(128KRAM) 약 26페이지

(3) 새로운 문서 작성(new word processor)

메인 메뉴에서 Add files to the desktop→ Add files에서 Make a new file for the word processor 를 선택한다.

① From scratch(새로운 파일 작성)

새로운 파일의 이름을 입력하고 리턴을 누르면 워드프로세서를 사용할 수 있다.

② From a text(ASCII) file(ProDOS로 포매팅된 디스크에 수록된 ASCII 텍스트 파일을 사용)

Apple Writer, PASCAL Editor 등으로 만든 텍스트 파일을 그대로 사용할 수 있다. 역시 파일 이름을 입력한다.

(4) 내용의 입력과 편집

① 리턴키의 사용

문단을 끝낼 때, 빈 줄을 만들 때, 구두점없이 줄을 끝낼 때(리턴키를 누르면 줄이 바뀐다) 사용된다.

ȯ-Z를 누르면 리턴이 사용된 위치를 표시해 준다.

② Word Wraparound(단어의 이동 재구성)

영어로 쓰여진 책들을 보면, 문장을 써내려가다가 줄끝에서 단어가 딱 맞지 않았을 때 하이픈(-)을 긋고, 계속 이어서 쓴다. 예 Existing Word Proce-ssor File

그러나 이 워드프로세서에서는 자동적으로 줄끝의 완전하지 못한 단어를 다음 줄의 맨 앞에 위치시켜 주므로 신경쓸 필요가 없다.

③ 사용키의 설명

ȯ(Open-Apple)과 함께 사용하기도 하고 단독으로도 사용한다.

i) ȯ C(copy)

텍스트 250줄 이하까지 복사할 수 있다. 원문은 지워지지 않는다.

· Within document : 복사할 부분을 화살표키로 정한 후 리턴을 누른다. 그런 후 옮길 위치에 화살표키로 커서를 놓고 리턴을 누른다.

· To Clipboard : 보조 메모리로 옮긴다.

· From Clipboard : 보조 메모리로부터 옮겨 온다. 보조 메모리는 To Clipboard로 다른 내용을 옮기기 전까지는 지워지지 않는다.

ii) ȯ-D(delete)

삭제할 부분에 커서를 놓고 ȯ-D한 후 커서를 이동시켜 리턴을 누른다. ESC는 취소할 때 쓴다.

iii) ȯ-F(Find)

문서의 내용을 찾아볼 수 있다.

· Text : 대 · 소문자를 구별하지 않고 모두 찾아낸다. 즉 GOOD, Good, good은 모두 같은 것으로 취급한다.

· Page : 지정한 페이지를 찾는데, 페이지 지정은 ♂-O(Option)로 할 수 있다.

· Marker : 지정한 Marker를 찾는데, Page처럼 ♂-O에서 설명하겠다.

· Case sensitive text : 대 · 소문자를 구별하여 찾는다. 즉 GOOD, Good, good은 모두 다른 문자로 취급한다.

· Options for printer : 코드에 의해 프린터 옵션을 찾는다. 자세한 것은 역시 ♂-O 에서 설명하겠다.

iv) ♂-K(calculate)

페이지 수를 계산한다.

v) ♂-M(Move)

원문이 지워진다는 것 외에는 Copy기능과 같으므로 설명을 생략한다.

vi) ♂-N(Name)

작성중인 파일의 이름을 바꾼다.

vii) ♂-O(Option)

이 기능은 문서의 포맷과 프린트되는 형식을 자유자재로 구성할 수 있다.

프린터 옵션을 사용하려면 설정하거나 변경하려는 옵션의 코드를 입력한다. 수치 입력시 한 자리수는 소수점이 필요없다. 끝낼 때는 ESC로 나온다.

· PW(Platen Width) : 프린터 용지의 폭
디폴트 = 8.0인치, 최대값 = 13.2인치

· LM(Left Margin) : 페이퍼 왼쪽 끝과의 간격
디폴트 = 0, 최대값 = 9.0인치

· RM(Right Margin) : 페이퍼 오른쪽 끝과의 간격
디폴트 = 0, 최대값 = 9.0인치

· CI(Chars per Inch) : 인치당 프린트되는 문자수(밀도 조정)
디폴트 = 인치당 10자, 가능한 값 = 4~24(프린터가 프린트할 수 없는 숫자를 입력하면 가장 최근의 값이 지정된다.)

· P1(Proportional-1) & P2(Proportional-2) : Data Base의 밀도 조정인 CI외에 워드프로세서는 P1, P2가 추가되었다. 단 P1과 P2는 CI와 같이 쓸 수 없다. P1과 P2는 문자에 따라 크기를 상대적으로 변화시키는 기능이다. 즉 CI를 설정하면 m과 i는 같은 넓이를 차지하지만 P1이나 P2는 m이 i보다 넓게 조정된다

· IN(INdent) : 처음 한 줄은 LM 지정 위치부터 시작하여 다음 줄부터는 지정한 스페이스만큼 안쪽으로 들어간다. 이때 첫줄에 O, * 또는 다른 캐릭터(character)를 표지로 붙일 수 있다. 0으로 지정하면 기능이 취소된다.

· JU(JUstified) : 오른쪽, 왼쪽 끝을 모두 맞춘다.

· UJ(UnJustified) : 왼쪽 끝만 맞춘다.

· CN(CeNtered) : 맞춘다.

· PL(Paper Length) : 사용하는 페이퍼의 수직 길이.
디폴트 = 11인치, 최대값 = 25.4인치

· TM(Top Margin) : 페이퍼 상단에서 프린트 첫줄까지의 거리.
디폴트 = 0, 최대값 = 9인치

주 트렉터가 없는 프린터는 TM을 0에 두고 페이퍼의 위치를 수동으로 조정해 주어야 한다.

· BM(Bottom Margin) : 프린트 끝줄에서 페이퍼 하단까지의 거리.
디폴트 = 2인치, 최대값 = 9인치

· LI(Lines per Inch) : 페이퍼 상하로 1인치당 프린트되는 줄의 수.
디폴트 = 6줄, 가능한 값 = 6~8

· SS(Single Space) & DS(Double Space) & TS(Triple Space) : 줄간격 옵션으로 한 줄(SS), 두 줄(DS), 세 줄(TS)로 지정한다.
디폴트 = SS

· NP(New Page) : 새로운 페이지를 지정한다. 직접 페이지를 지정하려면 새로운 페이지의 상단으로 하여 해당 줄에 커서를 놓고 ㅎ-O를 누른 후 NP를 입력한다. 타이틀이 새로운 페이지의 상단에 가게 할 수 있다. 사용자가 정한 페이지는 분할시키지 않지만 그 양이 한 페이지를 넘으면 분할한다. 문단의 가운데에서 새로운 페이지를 지정해주면 문단의 앞에서 분할하게 된다. 따라서 이런 경우에는 미리 그 위치에 CR(Carriage Return)을 넣어두어야 한다.

· GB(Group Begin) & GE(Group End) : 그룹 지정으로 특정한 내용을 하나의 그룹으로 묶어서 페이지 사이에서 나누어지지 않도록 한다.

GB : 그룹의 시작 위치 설정

GE : 그룹의 끝 위치 설정.

· HE(page HEader) & FO(page FOoter)

HE : 각 페이지의 상단에 한 줄의 Header를 넣는다. 본문과 두 줄 간격을 띄우게 된다.

FO : 각 페이지의 하단에 한 줄의 Footer를 넣는다. 텍스트 끝줄과 두 줄 이상을 띄우게 된다. 해제하고 싶으면 빈 줄로 지정하면 된다. Footer는 프린트되지 않는다.

· SK(SKip lines) : 그 지점에서 프린터가 비워두고 건너뛰어야 할 줄의 수를 정한다. 빈 공간을 남길 때 사용한다.

· PN(Page Number) : 페이지 수 부여. 프로그램이 정한 페이지를 무시하고 직접 페이지 수를 정해줄 수 있다. 페이지 수를 정하려면 PN을 입력한 후 부여하려는 페이지 수를 입력한다(최대 511). 그 다음의 모든 페이지는 자동적으로 페이지 수가 부여된다.

· SM(Set Marker) : Marker를 설정하는데 Marker의 번호는 1~254까지이다. 이것은 ㆁ-F에서 Marker 선택시에 이용된다.

· BB(Boldface Begin) & BE(Boldface End) : 프린트시 굵은 문자 출력에 이용된다. BB나 BE를 선택하면 ^가 표시된다. 내용을 알려면 커서를 그곳에 위치시키면 화면 하단에 나온다.

BB : Boldface(굵은 문자)의 시작

BE : Boldface의 끝

· + B(Superscript Begin) & + E(Superscript End)

+ B : Superscript(어깨글자, 기호, 숫자)의 시작, − E : Superscript의 끝

예 y + B2 + E = 4px⇒ y^2 = 4px

· −B(subscript begin) & −E(subscript end)

−B : Subscript(아래에 적은 문자 · 숫자)의 시작

−E : Subscript의 끝

예 C-B2-EH-B5-EOH⇒ C_2H_5OH

· UB(Underline Begin) & UE(Underline End) ┌ UB : Underline(밑줄)의 시작
└ UE : Underline의 끝

· PP(Print Page No.) : 페이지 수의 프린팅. 페이지 수는 어떤 것이든 프린팅시킬 수 있으며 Header, Footer, 텍스트 속에 넣을 수 있다. 원하는 위치에 커서를 놓고 페이지 수보다 단어를 우선시키려면 ESC로 프린터 옵션에서 빠져나온 후 원하는 위치에 페이지를 입력하고 스페이스바를 한 번 눌러 주고 ㆁ-O, PP 순으로 입력한다. 그러면 스페이스 사이에 ^ 가 삽입된다. 커서를 ^ 에 놓으면 Print Page No.가 화면 하단에 나온다.

· EK(Enter Keyboard) : 키보드 입력. 프린팅 도중에 프린터를 일단 중지시키고 키보드로부터 문서 속에 직접 내용을 입력할 수 있다.

원하는 위치에 커서를 두고 ㆁ-O, EK순으로 입력한다. 그 위치에 ^가 삽입되고 이때, 화면 아래에는 Enter Keyboard 메시지가 나온다. 입력한 후에 RETURN 하면 다시 프린팅을 시작한다.

iii) ㆁ-P(Print)

프린트를 시작한다.

ix) ⍥ -R(Replace)

문서 내용의 일부 또는 전부를 30자까지 지정하여 대체시킬 수 있다. 가장 최근에 대체(⍥-R) 또는 검색(⍥-F)한 텍스트는 기억되어 있어 RETURN하여 그대로 사용하든가, 새로 입력하여 준다. 지정한 텍스트가 없으면 Not found, press spacebar to continue라고 나온다.

x) ⍥ - T(Tab)

탭은 화면 위쪽의 두 점선에 세로줄로 표시되어 있다. 처음에는 다섯칸씩 되어 있으나 변경하거나 제거할 수 있다.

⍥-T를 누르면 커서가 화면 위의 두 점선으로 간다. 화살표키로 커서를 이동시켜 S(Set)로 탭을 설정하거나 C(Clear)로 제거한다. R(Remove)은 모든 탭을 제거한다. 빠져나올 때에는 ESC를 누른다.

xi) ⍥ -Z(Zoom)

문서에 사용된 RETURN의 위치와 프린터 옵션을 보여주거나 지워준다.

xii) ⍥ -space bar: sticky space 사용

앞뒤의 단어를 연결시켜서 한 단어처럼 사용할 수 있다. ^로 표시된다.

예 Phoebe Cates의 이름과 성 사이에 Sticky Space를 사용하면(Phoebe Cates) 줄의 끝에서 나누어질 경우 이름 전체가 다음 줄 맨 앞으로 이동한다.

xiii) CTRL-B(Boldface)

BB와 BE의 기능을 간단히 CTRL - B를 눌러서 사용할 수 있다.

xiv) Contrd-L(underline)

UB와 UE의 기능을 역시 CTRL - L로 간단히 사용할 수 있다.

xv) RETURN

앞에서 설명했으므로 생략하겠다.

xvi) DELETE

커서 왼쪽의 한 자를 삭제한다.

xvii) 화살표와 ⍥

화살표키는 커서 이동에 사용한다.

⍥ - →: 다음 단어로 이동

⍥ - ←: 바로 앞 단어로 이동

⍥ - ↑: 화면 상단으로 이동. 화면 상단에서는 앞 페이지로 이동

⍥ - ↓: 화면 하단으로 이동. 화면 하단에서는 다음 페이지로 이동

xviii) 탭과 ⍥

⍥-T로 수정하거나 원래 있던 탭만큼씩 이동한다.

TAB : 오른쪽으로 이동

 - TAB : 왼쪽으로 이동

(5) 참고 사항

① 프린터 옵션 사용시 디폴트값

-O를 눌렀을 때 중앙에 인버스(inberse)되어 있는 부분에 나와있다.

i) 좌우 간격

PW = 8.0인치

LM = 1.0인치

RM = 1.0인치

CI = 10자

ii) 배치

UJ(unjustified) : 왼쪽 끝은 같으나 오른쪽 끝은 줄마다 달라진다.

iii) 상하 간격

PL = 11.0인치

TM = 0.0인치

BM = 2.0인치

LI = 여섯 줄

iv) 줄간격

SS : 한 줄

② -F와 -R 사용시

i) 검색하거나 대체할 문자로 in을 입력하면 in, index, suggesting, son-in-law 등을 모두 찾는다. 따라서 찾으려는 문자 앞뒤에 스페이스를 넣어주면 찾고자 하는 것만(여기서는 in)을 찾을 수 있다.

ii) -R에서 설명하지 않은 것

One at a time : 첫번째 발견 위치에서 Yes, No를 묻는데 Yes로 답하면 대체되고 Find next occurrence?에 Yes로 답하면 계속된다. ESC로 빠져나올 수 있다.

③ -K

AppleWorks가 페이지 수를 계산할 때의 원칙은

i) 한 문단은 나누지 않는다. 한 페이지에서 새로운 문단이 시작하려면 최소한 두 줄이 있어야 하며 그렇지 않으면 다음 페이지로 넘어간다.

ii) 사용자가 지정한 것은 우선적으로 지켜진다.

iii) 나누지 않도록 지정된 것은 나누지 않는다.

파일에도 페이지 분할이 표시되며 다음에 문서를 보면 페이지가 바뀌는 위치에 점선으로 표시가 된다.

페이지 분할 이후에 내용을 변경시키면 페이지는 취소된다.

§7. 스프레드쉬트

(1) 스프레드쉬트의 기능

① Work with information in worksheets : 숫자 · 수식의 입력 · 수정 · 복사, 쉬트의 표시 방법 · 형태 변경.

② Format reports : 프린트 내용 지정, 원하는 형태의 포맷 작성.

(2) 파일의 사양

① 셀(cell)의 총수 : Column 127개 × Row 999개 = 126,873개

② 동시 사용 가능한 셀의 수 : 최대 1,000개(64K RAM), 최대 6,000개(128K RAM)

③ Column 번호 : A～Z, AA～AZ, BA～BZ, CA～CZ, DA～DW.

④ Row 번호 : 1～999

(3) 스프레드쉬트의 이해

스프레드쉬트는 행과 열에 따른 계산 작업을 쉽게 한다. 이 계산 작업은 결과의 예측과 직접 계산 상황을 보면서 할 수 있다.

(4) 새로운 스프레드쉬트 작성(new spreadsheet)

메인 메뉴에서 Add files to the desktop→ Add files에서 Make a new file for the spreadsheet를 선택한다.

① From Scratch(새로운 파일 작성)

새로운 쉬트를 만든다. 파일 이름을 입력하면 스프레드쉬트의 준비 화면이 나온다.

② From a DIF File

프로그램의 데이타베이스나 Visicalc 등의 다른 프로그램이 만든 DIF 파일을 사용할 수 있다. 단 Column방향으로 작성된 것이어야 한다. 선택 후 Pathname과 새로운 이름을 입력해야 한다.

③ From a Visicalc File

Visicalc의 자체 포맷 파일을 사용할 수 있다. 함수는 AppleWorks에 있는 것만 로드되고 일부 포매팅 명령은 로드되지 않는다. 수식은 75자까지이다.

(5) 스프레드쉬트 내의 사용키

- A(Arrange) : 내용의 배열
- B(Blank) : 쉬트의 내용 지우기
- C(Copy) : 내용의 복사
- D(Delete) : Column, Row의 삭제
- F(Find) : 특정한 셀이나 내용의 검색
- I(Insert) : Column, Row의 삽입
- J(Jump) : 윈도우의 커서 점프
- M(Move) : Column, Row의 이동
- N(Name) : 파일의 이름 변경
- O(Options) : 프린터 옵션
- P(Print) : 원하는 부분 프린트
- T(Titles) : 고정 타이틀의 설정
- U(Use) : 셀 수정
- W(Windows) : 쉬트의 분할
- Z(Zoom) : 수식의 표시

RETURN : 타이프한 내용의 입력
← ↓ ↑ → : 커서 이동
TAB과 - TAB : 좌우의 셀로 이동
 - 방향키(← ↓ ↑ →) : 커서를 상하좌우 끝으로 이동. 다시 한 화면 건너뛴다.
0~9 + − . : 수치(value)의 형태
" or Letters : 레이블(label)의 형태

(6) 내용의 입력과 수정

① 입력
· CTRL - E(삽입 · 삭제 커서) : 셀에 내용 타이프
· DELETE(DEL키) : 커서 왼쪽의 한 자 삭제
· RETURN & 방향키
② 수정

수정하려는 셀에 커서를 놓고 -U (Use edit feature)를 누르면 셀의 내용이 화면 하단에 나온다. 이때 다음과 같이 수정한다.

· ESC : 원래 엔트리(entry)를 회복시키고 커서를 원래 위치에 놓는다.

· ←, → : 커서 이동

· CTRL-Y : 커서 위치에서 엔트리 끝까지 삭제.

수정이 끝나면 리턴한다. 그러면 프로그램은 에러를 점검하여 문제가 있으면 삑 소리와 함께 정정을 기다린다.

(7) 레이블의 사용

수치가 없는 레이블에 사용되며 문자로 시작하거나 "로 시작한다. 레이블이 셀보다 크면 그 옆의 셀에 자동적으로 나누어 넣는다. Auto Repeat 기능을 이용하여 입력한 반복형 레이블은 Repeated라고 표시되며, -U로 수정할 수 없다.

(8) 수치의 사용

숫자, 포인터, 수식, 함수가 포함된 엔트리이다.

① 숫자(numeral or figure) : 0~9, +, – 또는.(소수점)으로 시작한다.

② 포인터(pointer) : 쉬트 내의 다른 셀을 지정하는 것이다. 레이블이 아님을 알리기 위해 반드시 +나 –로 시작해야 한다. +, –, 소수점 입력 후에는 화살표키로 셀을 지정할 수 있다.

③ 수식(nemerical formula) : 계산을 위한 공식으로서 그 결과만 화면에 표시된다. 숫자, 산술 기호, 포인터, 함수가 두 개 이상 들어가며 반드시 +나 –로 시작해야 한다. +, –, 소수점 입력 후에는 화살표키로 셀을 지정할 수 있다.

④ 수식(numerical formula) : 계산을 위한 공식으로서 그 결과만 화면에 표시된다. 숫자, 산술 기호, 포인터, 함수가 두 개 이상 들어가며 반드시 +나 –, 소수점, 숫자, 괄호, @로 시작해야 한다. 역시 셀 지정에 화살표를 이용할 수 있다.

⑤ 함수(function) : 일정한 계산 과정을 표시하는 코드이다. 반드시 @로 시작하여 그 뒤에 다음과 같은 함수의 약자를 쓴다.

· 산술함수 : @ ABS(ABSolute), @ AVG(AVeraGe), @ CHOOSE, @ COUNT, @ ERROR

@ INT(INTeger), @ LOOKUP, @ MAX(MAXimum), @ MIN(MINimum), @ NA, @ SQRT (SQuare RooT), @ SUM

· 회계함수 : @ NPV

· 논리함수 : @ IF

(9) 스프레드쉬트 표준 수치

쉬트의 화면 표시 방법을 지정하는 것으로 언제든지 ♂-V 기능을 이용하여 변경시킬 수 있다. ♂-L 기능을 이용하면 특정한 셀의 표준 수치만 바꿀 수 있다.

① 수치의 포맷

- Fixed : 0~7까지 소수점 이하의 자리수를 지정한 만큼 표시. 예 -137.00
- Dollars : 앞에 $표시. 예 $ 113.58
- Commas : 1000단위마다 표시. 음수는 괄호 안에, 정한 숫자만큼의 소수점 이하를 표시.
  예 1,436.04
- Percent : 셀의 내용에 100을 곱하여 정해진 수의 소수점 이하(0~7)를 표시. 예 1234.5 %
- Appropriate : 입력하는 대로 표시.

② 레이블의 포맷

- Left justified(왼쪽 배치)
- Right justified(오른쪽 배치)
- Centered(중앙 배치)

③ 컬럼 넓이

표준은 아홉 자이지만 1~75까지 변경할 수 있다.

④ 프로텍트(protect)

엔트리의 내용을 보호한다. 보호는 ♂-L, 해체는 ♂-V를 사용한다.

⑤ 재계산

- 순서 설정 : Column 우선, Row 우선
- 시기 설정 : Automatic(자동), Manual(수동)

새로운 값이 입력될 때 그에 관련된 계산을 다시 하는 것이 재계산이다. 수동으로 설정하면 ♂-K를 눌렀을 때만 재계산을 행한다.

⑥ 표준 수치의 디폴트

- 수치 포맷 : Appropriate
- 레이블 포맷 : Left Justified
- 컬럼 넓이 : 아홉 자
- 프로텍트 : 기능 작동
- 재계산 : Column 우선, 자동. 컬럼 가장 위의 레이블은 표준 수치에 영향을 받지 않도록 되어 있다. 프로텍트는 직접 표시되지 않는다.

⑦ 표준 수치의 변경

♂- V를 누른 후 변경시키려는 대상에 따라서 선택해 나간다. 단 컬럼 넓이의 변경은 ♂ - →, ←로

변경시키고 ESC로 끝낸다.

(10) 셀 구조의 작업

특정한 한 개 또는 몇 개의 셀만 표준 수치와 다르게 표시되도록 한다.

-L을 누른 후 Entry(셀 한 개), Rows · Columns(한 줄의 여러 셀), Block(한 구역을 지정) 중에 선택하고 화살표키로 대상 셀들을 반전시키고 리턴한다.

이후의 진행은 같다. 다만 Standard Option(표준 옵션)은 쉬트의 표준 수치로 되돌아가며 프로텍트의 옵션은 다음과 같다.

Lables only– 레이블만 재입력 가능

Valuse only– 수치만 재입력 가능

Nothing– 재입력 불가능

Anything– 프로텍트

(11) 스프레드쉬트의 리포트 작성

리포트 작성시에는 미리 포함시킬 내용과 리포트의 넓이를 결정해 두어야 한다.

① - P를 눌러 All, Rows, Columns, Block 중에서 프린트 범위를 선택한다.

② Rows, Columns, Block 선택시에는 커서를 이용하여 프린트하려는 부분을 반전 표시한다.

③ 프린트 메뉴의 내용을 점검하여 리포트의 넓이가 적절한가를 본다. 너무 넓으면 ESC 를 눌러 Review/ Add/Change 모드로 돌아가 리포트를 재설계한다.

(12) 프린터 옵션의 사용

- O를 누르면 프린터 옵션 화면이 나온다.

이 스프레드쉬트의 프린터 옵션은 데이타베이스와 워드프로세서에서와 같이 설정이나 변경 방법이 같으므로 데이타베이스와 워드프로세서의 프린터 옵션을 참고하기 바란다.

§ 8. Cut & Paste 기능의 사용 방법

Clipboard를 사용하는 기능으로서 그 가치는 매우 높다. Clipboard에는 최대 250개의 자료를 보관할 수 있다. Cut 기능은 파일 내의 레코드를 일시 보관하는 기능이고 Paste 기능은 보관했었던 레코드를 메모리에서 빼와서 다른 파일 내에 재입력시키는 중요한 기능이다. 전송, 이동 방법에는 두 가지가 있다.

이동 : Review/Add/Change 모드에서 - M 사용이 바로 이것인데 원래의 레코드가 없어진다

는 단점이 있다.

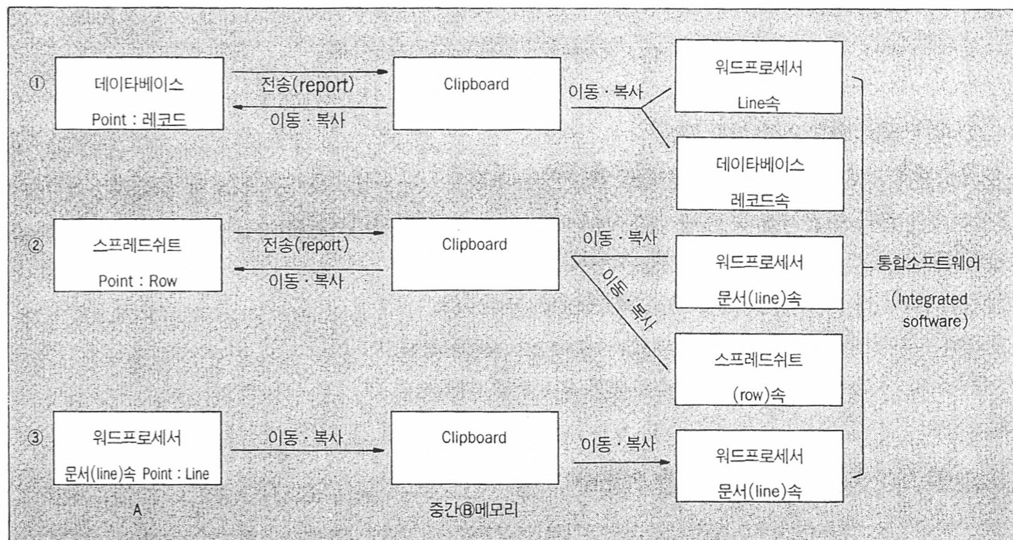
복사 : Review/Add/Change 모드에서 ♂- C 사용이 바로 이것인데 ♂- M의 단점을 보완한 기능이다.

AppleWorks V2.0이 통합 소프트웨어로 인기를 끄는 점도 바로 이 Cut & Paste 기능 덕분이다. 예를 들면 데이타베이스나 스프레드쉬트의 자료를 Clipboard로 옮긴 후 이것을 다시 워드프로세서 문서 속으로 이동 · 복사시킬 수 있는 기능인 것이다.

이 때에 전송되는 중심을 살펴보면
- 스프레드쉬트 : Row
- 워드프로세서 : 줄(Line)
- 데이타베이스 : 레코드

로 이동 · 복사된다.

Cut & Paste 기능은 중요하므로 도표를 그려보았다.



여기서 ①번의 그림을 살펴보면 스프레드쉬트로의 전송은 활용될 수 없다. 이 단점은 호환성이 좋은 DIF 파일을 사용하는 것으로 보충할 수 있다. 참고로 데스크탑과 Clipboard와의 메모리 영역들은 아무런 관계가 없다.

그림 8.5 Cut & Paste 기능

1. 데이타베이스에서의 Cut & Paste 기능

전송될 수 있는 중심은 레코드 위주이며 한 번에 250개의 레코드를 전송 · 복사할 수 있다. 같은 데이타베이스끼리와의 전송은 완전한 형태의 레코드만이 가능하고 전송은 리포트로 편집되어야만 이루어질 수 있다.

(1) 데이타베이스끼리와의 전송

① Review/Add/Change 모드에서 화면을 다중 구조로 입력해 둔다.

㈜ 단일 구조에서는 전송이 이루어지질 못한다.

② 커서의 위치를 전송시키려는 엔트리의 첫자에 위치해 둔다.

③ ⌘- M과 ⌘- C를 사용하여 복사. 커서키를 사용하여 이동하려는 레코드를 반전시키고 리턴하면 복사된다. 물론 이때의 기능은 Clipboard의 입력, 즉 "To Clipboard"이어야만 한다.

④ ⌘-Q를 선택, 데스크탑의 목록을 살피고 알맞은 파일을 로드한다.

⑤ 새 파일을 Review/Add/Change 모드에서 다중 구조로 입력해 둔다.

㈜ 단일 구조에서는 전송이 이루어지지 않는다.

⑥ "From Clipboard"를 사용하여 이동 · 복사를 실시한다.

㈜ 특정 레코드의 선택 방법은 ⌘- F로 하든가 ⌘- R로 선정 기준을 두어서 진행시키면 된다.
이동 · 복사의 유용성을 위하여 한 개 정도의 빈 레코드가 있는 파일이 있으면 좋다.

(2) 워드프로세서 문서와의 전송

① ⌘- P를 눌러 데이타베이스 리포트 형식으로 바꾼다.

② 프린트 대상의 리포트를 전과 같은 방법으로 Clipboard에 입력시킨다.

③ "Header"가 있을 때에는 날짜를 입력해 준다.

④ ⌘- Q로 데스크탑의 목록을 살핀다. 파일을 로딩한다.

⑤ 워드프로세서 속의 문서에서 ⌘- M, ⌘- C를 입력시킨다.

⑥ "From Clipboard"를 사용하여 전송된 리포트를 복사한다.

2. 스프레드쉬트에서의 Cut & Paste 기능

작성된 파일을 다른 스프레드쉬트의 파일 속이나 워드프로세서의 문서 속에서 리포트를 작성하여 이동 · 복사할 수 있다.

한 번에 250개의 Row를 전송시킬 수 있다.

(1) 스프레드쉬트에서의 전송

① 파일을 로드→ Review/Add/Change 모드에서 전송하려는 Row를 커서로 위치해 놓고 ⌘- M이나 ⌘- C를 이용, 선택한다.

② ⌘- Q를 선택, 데스크탑의 파일들 중에서 골라서 로딩한다.

③ ⌘- M, ⌘- C를 누른 후 From Clipboard를 선택, 전송한다.

(2) 워드프로세서 문서와의 전송

① ♂- P를 선택, 스프레드쉬트 자료를 리포트 형식으로 변환시킨다.

② All, Rows, Columns, Block을 선택하여 전송하려는 중심을 입력한다.

③ 커서키를 이용, 전송하려는 파일들을 반전시킨다.

④ "Header" 입력 요구시에 날짜를 입력해 준다.

⑤ ♂- Q로 파일들을 리스트해 보고 선택해 준다.

⑥ ♂- M, ♂- C를 삽입하려는 위치에서 From Clipboard를 사용, 파일을 뽑아낸다.

주 • ♂- M과 ♂- C의 차이

♂- M	♂- C
약간의 메모리를 사용한다.	더 많은 메모리를 차지한다.

리포트의 최대 넓이는 225자이다.

참고로 전송된 모든 파일들은 그 프로그램의 기능을 전부 사용할 수 있다.

3. 워드프로세서 문서 속에서의 Cut & Paste 기능

워드프로세서의 전송에서는 워드프로세서 파일들끼리만이 사용 가능하다. 한 번에 전송될 수 있는 워드프로세서의 라인은 모두 250줄이다.

(1) 워드프로세서 문서와의 전송

① 전송할 문서 속의 라인의 첫자나 끝자에 커서를 위치해 놓는다.

② 전송에 쓰이는 키는 역시 ♂- M, ♂- C를 사용, Clipboard로 입력시킨다.

③ 커서키를 사용하여 전송하려는 라인들을 반전시켜 준다. 리턴하면 전송된다.

④ ♂- Q를 선택, 데스크탑의 파일을 살펴본다.

⑤ 전송시키려는(COPY) 곳에서 ♂- C, ♂- M을 선택 "From Clipboard"를 선택한다.

§9. 프린터 출력에 관련되는 기능의 사용법

• 데이타베이스와 워드프로세서와 스프레드쉬트 등으로 사용하여 만든 모든 파일을 프린팅할 수 있다.

• 데이타베이스와 스프레드쉬트 등에서 만든 리포트는 Clipboard를 이용해 워드프로세서 속에 입력하여 다시 프린팅할 수 있다.

• 언제든지 AppleWorks의 화면은 ♂-H를 이용, 하드 카피할 수 있다.

• 리포트나 문서의 프린팅을 위한 프린터 옵션이 미리 설정되어 있어야 하며 프린터의 종류가 이미

지정되어 있어야만 프린팅이 제대로 이루어질 것이다.

• 문서나 리포트의 파일들이 디스켓에 수록될 때에는 그 프린트 옵션이나 프린트 코드가 함께 수록되어 입력된다. 따라서 나중에 프린팅할 때에는 미리 입력되었던 프린터로만 사용 가능하다.

• 데이타베이스와 스프레드쉬트 파일들의 리포트를 DIF 파일로 디스크에 세이브시킬 수 있다. 그리고 이 파일을 다른 프로그램에서의 소스 파일로 이용할 수가 있다. DIF 파일을 프린팅할 때에는 역시 새로운 파일의 이름을 정해주어야 한다.

1. AppleWorks에서 사용 가능한 프린터들

어느 프린터든지 사용할 수 있지만 프로그램마다 설정되어 있는 기기들이 틀리기 때문에 이때마다 프린터의 종류를 알맞게 선택해 주어야 한다.

AppleWorks에서 사용될 수 있는 프린터들

- Apple Imagewriter
- Apple Silentype
- Apple Daisy Wheel Printer(DWP)
- Epson MX Series
- Epson MX/Graftrax+
- Epson RX Series
- Epson FX Series
- Qume Sprint 5
- Qume Sprint 11
- Apple Scribe (구형 AppleWorks에는 수록되어 있지 않다.)

㈜ 리스트에 없는 프린터일 경우에는 "Custom Printer"를 사용, 직접 프린터의 옵션을 설정해 주어야 한다.

2. 프린터의 설정 방법

메인 메뉴에서 Other Activities를 선택하고, Specify information about your printer(s)를 선택하여 설정 화면으로 들어간다.

• Open-Apple-H printer : 하드카피할 경우에 쓰이는 기능으로 선택된 뒤 현재 입력 또는 추가된 프린터의 종류에서 사용할 프린터를 고른다.

• Add a printer(maximum of 3) : 최대 세 개의 프린터의 옵션들을 입력시켜 놓을 수가 있다. 만약 현재 입력되어 있는 프린터의 갯수가 세 개라면 프린터의 삭제를 요구할 것이다.

• Custom printer : 리스트에 없는 프린터를 직접 조정하는 기능으로서, 선택되면 사용자의 임의대로 프린터의 이름을 타이핑한다. 잠시 후 슬롯 번호나 디스크를 묻는 항목이 나오는데, 알맞은 번호를 입력해 주고 리턴해 주면 다음과 같은 여섯 가지 옵션이 나온다.

① Needs line feed after each RETURN : 프린팅 실시 후 라인피드가 자동적으로 잘 이루어지지 않는 프린터이거나 수동적으로 프린팅될 경우에는 이 항목을 변경시켜 보기 바란다. 방법은 Yes 와 No 로 선택한다.

② Accepts- top- of- page commands : 이 기능은 프린트하는 도중에 프린트되고 나면 다음 페이지 상단으로 Head가 옮겨지는 기능을 가진 프린터에만 쓰일 수 있는 기능이다. 일종의 "Form Feed" 라 한다. 역시 Yes, No로 답해준다.

③ Stop at end of each page : 한 페이지의 프린팅이 끝날 때마다 프린터를 정지하게 하여 용지를 갈아끼울 수 있게 하거나 일부 삭제하게 해주는 기능이다. Yes, No로 답한다.

④ Platen width : 프린팅의 간격으로 각각의 리포트에 지정된 수치와 같거나 더 넓어야 한다. 수치의 값을 입력해야 한다.

⑤ Interface Cards : 이 기능은 흔히 쓰이고, 있는 센트로닉스 방식의 패럴렐 인터페이스를 사용한 프린터에서도 프린팅이 가능케 해주는 기능을 담당하고 있다. 가지고 있는 카드의 이름을 입력해 주어야 한다.

⑥ Printer Codes : 프린터에 모양이나 줄의 수를 입력하여 프린트하게 하는 기능으로서 ←, →, 리턴키를 사용한다.

• Characters per inch : 한 인치당 프린팅할 수 있는 문자 수를 설정해줄 수 있는 키나 컨트롤 코드로 입력해 준다(예 : '리턴'을 쳐주면 리턴키 자체가 컨트롤키가 되는 것이다). 입력이 끝나면 SHIFT-6을 누르고 더 컨트롤 키를 정할 때에는 ESC 키를 사용한다.

㈜ 소수점 사용 불능.

프린터의 능력에 맞는 기능을 부여하여야 함.

인치당 문자 수에 입력을 하지 않을 경우에는 프린트시 텍스트 표시가 나타남.

• Lines per inch : 인치당 줄의 수를 지정하는 코드를 설정할 수 있다. 범위는 6~8이다.

• Boldface, Subscript, Superscript : 프린팅시에 인쇄되는 문자의 특수 코드를 정할 수가 있다.

㈜ 시작과 끝을 알리는 코드 값을 입력시켜야만 한다.

• Underlining : 프린팅시 글자마다에 언더라인을 쳐주는 기능이다. 언더라인 옵션에서는 다시 네 개의 선택 옵션으로 나누어진다. 이 옵션은 프린터가 언더라인을 해줄 수 있게 하는 부수적 기능이다.

Printer has start/stop underline commands : 프린터 기능에 언더라인 옵션이 가능할 때 이 항목을 선택한다. 언더라인의 명령의 조절은 시작과 끝의 컨트롤 키에 의해 조절된다.

Print character, backspace, underline : 언더라인 기능을 보유하고 있지 않은 프린터일 경우에

이 기능을 선택한다. 이 옵션이 선택되면 프린트되는 형식이 약간 달라진다. 한 글자마다 언더라인을 쳐가면서 되돌아가는 형식으로 프린팅된다.

Print line, Carriage return, Print underline : 언더라인 기능과 백스페이스 기능이 모두 없을 때 이 항목을 선택한다.

세 개의 기능들 중 한 가지라도 안될 때에는 포기한다.

3. 프린트 방법

(1) 데이타베이스에서의 리포트 프린팅, 디스크로의 프린팅

① ó-P를 선택, 어느 곳에 프린트할 것인가를 정한다. 이때의 형식은 '리포트'이다.

② 날짜를 입력한다.

③ 프린터의 선택과 디스켓의 선택을 숫자로 입력시켜 준다.

④ 텍스트 파일, DIF 파일, 디스크로의 편집(= save)을 선택하고 파일의 새로운 Pathname을 타이핑해 준다. ESC (save(= 프린팅)의 진행을 막을 수 있다)키는 Review/Add/Change 모드로 복귀한다.

(2) 워드프로세서에서의 리포트 프린팅

① Review/Add/Change 모드에서 ó-P를 선택, 시작한다.

- Begining : 현재 출력된 문서의 처음부터 프린팅
- This page : 현재 커서가 위치한 페이지 처음부터 프린팅
- Cursor position : 현재 위치한 커서부터 프린팅

이 기능들 중 한 가지를 선택한 후 프린터를 선택하면 프린팅된다.

ESC : 프린트하고 싶은 부분까지만 프린팅을 하거나 제동을 걸고 싶을 때.

② 페이지에서의 프린트 정지 방법은 프린트를 정지하고 싶은 다음 줄의 첫 공간에 커서를 위치해 놓고 ó- O를 선택, 'PE'를 변환시키고 리턴하면 된다 . ESC 키를 사용하면 다시 프린팅이 이루어진다.

㊟ 적당한 위치에서 ó-O를 사용, 'PH'의 값을 바꾸어주면 특정 페이지의 프린팅에서도 정지할 수가 있다.

(3) 스프레드쉬트의 리포트 프린팅

① Review/Add/Change 모드에서 커서를 프린트할 위치에 정해둔다.

② ó-P를 선택, All, Rows, Columns, Block중에 선택한다.

③ 커서키를 이용하여 원하는 부분만을 반전시킨다. 리턴한다.

④ Platen보다 리포트의 길이가 더 길면 ♂- O를 선택, 리포트의 인치당 문자수를 늘린다. 어느 정도 기능이 맞으면 리턴하고 프린팅을 실시한다.

⑤ 프린트할 장수를 입력한다.

⑥ 날짜를 입력한다.

㊟ 텍스트, DIF, 디스크로의 Save시(= Printing)는 Pathname을 쳐주어야 한다.

⑦ 프린팅을 실시한다(리턴으로…).

§ 10. 맺음말

이상으로 AppleWorks V2.0의 분석을 마치기로 한다. 이 패키지는 강력한 통합 소프트웨어 패키지로서 충분한 능력을 제공하지만 약간의 제약(한글)이 따른다. 하지만 앞에서 설명된 바와 같이 편리한 편집 기능과 처리 기능은 다른 패키지에 비해 월등히 뛰어나며 그 뛰어난 편집 능력은 프린터를 보유하고 있는 유저들에게는 최대의 충족감을 주었으리라고 확신한다. 뜻이 있는 유저들이 이런 좋은 유틸리티에 한번쯤 도전을 해 보아서, AppleWorks V2.0이 독자들에게 가장 인기있는 유틸리티가 되었으면 하는 마음 간절하다.

〈참고서적〉

ProDos User's Manual
ProDos Technical Reference Manual
AppleWorks Tutorial Manual
Using AppleWorks Manual

제 9 장

애플 IIe용 언어 프로그램

§ 1. Merlin-Pro
§ 2. Z-BASIC

§ 1. Merlin-Pro

애플의 어셈블러중 사용이 편리하다는 점과 65C02의 명령어를 사용할 수 있다는 장점을 가진 어셈블러가 바로 Merlin-Pro이다. Merlin-Pro는 애플Ⅱe용 어셈블러이기 때문에 Ⅱe의 공식 OS인 ProDOS하에서 작동한다. 그러므로 ProDOS의 사용법도 알아둘 필요가 있다. 그리고 Merlin을 사용해본 사람은 Merlin-Pro를 쉽게 조작할 수 있을 것이다.

어셈블러	Merlin-Pro	LISA V2.5	Tool Kit
전체적 등급	A	B	B^+
기 능	A	B^+	A
사용하기 쉬움	A^-	B^-	B
소스 파일의 종류	Textfile	Binary Textfile	Textfile
EXEC의 사용 가능	No	Yes	No
유용한 서브루틴	약 간	많 다	없 다
16K 램 카드	꼭 필요하다	없어도 된다	없어도 된다
Tab 기능	Yes	No	Yes
검색 기능	Yes	No	Yes
Local / Global 치환 기능	Yes	No	Yes
Copy Move 기능	Yes	No	Yes
문법 검사	어셈블할 때 한다	입력할 때	어셈블할 때
어셈블 속도	Fast	Fast !	Slow
표준 문법인가?	표 준	약간 차이가 있다	표 준
Immediate Operand의 종류	HEX Decimal CHR BIN OCT MSB LSB	HDCBML	HDCBOML
곱하기와 나누기를 식에 사용	Yes	Yes	No
Default Base	10진	16진	10진
Pseudo Op Code의 수	27	30	24
스트링 처리 Op Code	약 간	많 다	적 다
매크로 기능	Yes	No	No
조건부 어셈블 기능	Yes	No	Yes
Rero Catable Object Code	No	No	Yes
어셈블할 때 Object Code를 저장시키는 방법	디스크나 메모리에	메모리에만	디스크나 테이프에
소스 리스트의 최대 크기	메모리 어셈블 디스크 어셈블 : 무한 Pro-41K	메모리어셈블 : 26K (16K 램카드 사용시)	디스크어셈블 : 무한
Op Code를 토큰화시키는가?	No	Yes	No

표 9. 1 Merlin, LISA, DOS Tool Kit의 비교

표 9. 1은 Merlin과 LISA, DOS Tool Kit를 비교한 것이다.

1. 메인 메뉴

Merlin-Pro를 부팅시키면 메인 메뉴가 나온다. 선택은 원하는 기능의 첫글자를 누르면 실행된다.

Catalog

현 드라이브의 파일들을 보여준다. 그 때 Pathname을 물어온다. 여기서 그냥 리턴을 치면 현재 Prefix 되어 있는 것의 카탈로그를 보여준다.

Load Source

소스 파일을 메모리로 읽어들인다. 처음 로드할 때는 파일명을 직접 입력하고 다시 로드하거나 세이브할 때에 파일명이 처음의 것과 같으면 다시 입력할 필요없이 'Y'를 치면 된다.

로드된 후 자동으로 ED/ASM 모드로 들어간다.

Save Source

메모리의 소스를 디스크에 세이브한다.

Append File

새로운 소스 파일을 읽어서 현재 메모리에 있는 소스의 뒤에 추가한다. 매번 반복 사용되는 내용이라면 따로 파일을 만들고 이 기능을 이용해서 추가하면 편리할 것이다.

Disk Command

디스크 명령어를 처리하는 기능으로 서브디렉토리를 Prefix시켜 줄 때도 사용된다.

Enter ED / ASM

ED/ASM 모드로 들어간다.

Save Object Code

ED/ASM 모드에서 어셈블한 후 메인 메뉴로 빠져나와 어셈블된 목적 코드(object code)를 디스크에 세이브시킬 때 사용된다.

Set Date

파일을 디스크에 세이브할 때의 시간, 날짜 등을 입력한다.

Quit

Merlin-Pro에서 빠져나와 다른 ProDOS를 부트시킨다.

2. ED / ASM 모드

① 편집 명령

L(ist)

메모리에 있는 소스 리스트.

L

소스 전체 내용의 리스트.

L30

30번 행의 내용을 리스트.

L40, 70

40번부터 70번까지의 리스트.

L40 / 70

40번과 70번만 리스트.

. (period)

이전의 리스트 내용을 다시 리스트.

/ (slash)

현재 리스트된 다음의 번호부터 리스트.

이 명령어가 실행될 때 스페이스바를 누르면 리스트가 중지되고 다시 누르면 한 행씩 리스트된다. 계속 리스트하려면 ESC키를 누르면 된다.

A(dd)

현재 메모리에 새로운 내용을 추가할 때 사용한다.

처음에 이 명령을 사용하면 1번 행부터 입력이 시작된다. 입력할 때 스페이스바를 누르면 자동으로 TAB이 맞추어지므로 입력이 편리하다. 그 행의 입력이 완료되면 리턴키를 누르고 다음 행을 입력한다. 모든 내용을 입력하였거나 중간에 끝내려면 리턴키를 누르면 된다.

I(nsert)

지정하는 행번호 바로 전에 새로운 내용을 삽입시킨다. 이때도 A(dd)의 사용 때와 마찬가지로 내용을 다 입력한 뒤 리턴키만 누르면 빠져나온다.

예 : L

```
  1 ORG   $6000
  2 LDA   #$01
  3 LDA   #$AF
:I2
2 STA   $300
3 STA   $301
4 ↓
```

```
:L
1 ORG  $6000
2 STA  $300
3 STA  $301
4 LDA  #$01
5 LDA  #$AF
```

D(elete)

리스트중 원하는 행번호를 지운다.

예 D3 : 3번 행을 지운다.

D1/5 : 1번 행과 5번 행을 지운다.

D/1, 6 : 1번 행에서 6번 행까지 지운다.

D 2, 4/6/9, 10 : 예를 들어 D1/1/1과 같이 명령어를 내리면 원래의 내용 중에는 1, 2, 3번을 지우는 결과이다.

E(dit)

리스트 중의 내용을 수정한다.

예 E3 : 3번 행을 수정한다.

E : 전체 내용을 수정한다.

E 2, 5 : 2번 행부터 5번 행까지의 내용을 수정한다.

E 2/5 : 2번 행과 5번 행을 수정한다.

E. STA : STA란 내용이 들어있는 행을 수정한다.

E3, 6. LDA : 3번 행부터 6번 행까지의 LDA가 들어있는 행을 수정한다.

위의 기능을 이용해 에디터 모드로 들어가게 되는데 이 상태에서 사용할 수 있는 키들은 다음과 같다.

CTRL－I : 삽입 모드가 된다. 이후에 문자를 입력하면 커서 뒤의 문자부터 뒤로 밀려서 입력된다.

CTRL－D : 커서 위치의 한 글자를 지운다.

DEL : 커서 앞의 한 글자를 지운다.

CTRL－X : 현재 행의 수정을 취소하고 다음 행으로 간다.

CTRL－C : 수정 기능을 취소한다.

CTRL-Q : 커서 뒤쪽의 내용을 지우고 다음 행으로 간다.

CTRL－R : 현재 수정했던 내용을 원래의 내용으로 되돌린다.

CTRL－F : 문자 검색 모드로 이 키를 누르고 검색을 원하는 문자를 누르면 수정 중인 행 내의 그 문자의 위치로 커서가 움직인다.

CTRL－E : 그 행번호의 끝으로 옮긴다.

CTRL－P : 이 키를 누르면 *가 한번에 32개가 프린트되어 주석문을 달 때에 도움을 준다.

R(eplace)

Delete 기능과 Insert 기능을 합친 기능으로 지정하는 범위의 내용을 지우고 그 자리에 새로운 내용을 대치시킬 수 있도록 삽입 모드로 들어간다.

예 R5 : 5번 행을 지우고 5번 행부터 삽입 모드로 들어간다.

R3, 5 : 3번 행부터 5번 행까지 지우고 3번 행부터 삽입 모드로 들어간다.

C(hange)

지정한 범위 내에서 원하는 문자열을 찾아서 다른 내용으로 대치시킨다.

예 C.STA. LDA : 리스트 내에서 STA를 찾아서 LDA로 바꾼다.

C5, 10, STA. LDA : 리스트 내에서 5번과 10번 사이의 STA를 찾아서 LDA로 바꾼다.

이 명령을 실행시키면 'ALL OR SOME(A/S)?과 같은 질문이 나오는데 "A"를 입력하면 모든 경우를 치환시키고 "S"를 입력하면 찾는 문자열이 발견될 때마다 화면에 표시되고 바꿀 것인지를 묻는데 여기서 스페이스바를 치면 내용이 바뀌고 그밖의 키를 치면 다음의 경우를 찾는다.

F(ind)

지정한 범위 내에서 원하는 문자열이 있는 행을 찾아낸다.

예 F. LDA : 리스트 내에 LDA가 있는 행을 찾아서 리스트한다.

F 10, 15.STA : 리스트 내에 10번 행부터 15번 행까지 STA가 있는 행을 찾아 리스트한다.

COPY

지정한 범위의 내용을 다른 곳으로 복사한다. 이때 원래의 내용은 그대로 남아있다.

예 COPY 5 TO 10 : 5번 행의 내용을 10번 행의 앞에 삽입한다.

COPY 3,6 TO 10 : 3번 행부터 6번 행을 10번 행의 앞에 삽입한다.

MOVE

지정한 범위의 내용을 새로운 위치로 이동시킨다.

예 MOVE 5 TO 10 : 5번 행의 내용을 10번 위치로 이동시킨다.

MOVE 5, 10 TO 15 : 5번 행부터 10번 행까지의 내용을 15번 행의 위치로 이동시킨다.

NEW

현재 메모리에 있는 소스의 내용을 모두 지운다.

LEN(gth)

현재 소스가 몇 바이트가 사용됐고 몇 바이트가 남았는지를 보여준다. NEW한 후에 'LEN' 해보면 약 4KB, 즉 41215바이트가 남아 있음을 알 수 있다.

② 어셈블리 명령어

ASM

메모리에 있는 소스 리스트를 어셈블한다.

이 명령을 실행시키면 'UPDATE SOURCE(Y/N)?'라는 질문이 나오는데 이것은 소스의 내용 중에 프로그램한 날짜가 들어있을 경우 어셈블할 때마다 그 날짜를 고칠 수 있게 해준다. 이 때 'N'을 누르면 그냥 어셈블되지만 'Y'를 치면 소스의 내용 중 '/'에 의해 숫자가 분리된 부분을 찾아 수정할 수 있게 해주고 숫자를 고친 후 사용자가 리턴키를 누르면 어셈블이 시작된다.

VAL()

이 기능은 소스의 내용 중에 있는 레이블(label)이 어셈블될 때 어드레스값이나 문자의 아스키값을 알려준다.

예 VAL(START) : 레이블 'START'가 어셈블되었을 때의 어드레스를 알려준다.
VAL("B"):B의 아스키값을 알려준다.

MON(ITOR)

이 명령어를 사용하면 에디터에서 빠져나와 모니터 모드로 들어가는 데 필요한 일을 하고 CTRL-C를 누르면 Merlin-Pro의 메뉴로 돌아간다. 이 모니터는 65C02의 모니터이므로 65C02만의 OP코드를 볼 수 있을 것이다.

③ 그밖의 명령어

TRON(TRunc ON)과 TROFF(TRune OFF)

어셈블러가 4컬럼으로 맞추어져 있을 때에 Label, Op Code, Operand 뒤에 코멘트를 화면에 나오게 하거나 안나오게 하는 데 쓰인다.

TEXT

이 명령어를 사용하면 리스트할 때 탭이 빈 칸 하나로 대치되어 모든 내용이 밀집되어 나타난다.

PR# or VIDEO

출력을 다른 슬롯으로 보낼 때 사용한다. 예를 들어 어셈블 리스트를 프린터로 보내려면 PR#1이나 VIDEO 1을 하면 된다.

FIX

이 명령어를 이용하면 TEXT에 의해 밀집된 소스 리스트가 원래대로 고쳐져 TAB이 복구된다.

P(rint)

List는 소스의 내용이 행번호와 함께 나오는 데 반해서 Print는 행번호 없이 소스만 리스트된다.

TABS

Op Code 영역이 시작하는 컬럼 번호, Operand가 시작하는 컬럼 번호, 코멘트가 시작하는 컬럼 번호를 지정한다.

예 TABS 10, 14, 24

Op Code Operand 코멘트

USER

이는 Op Code를 사용자가 정의해서 사용할 수 있다.

Q

ED/ASM을 끝내고 메인 메뉴로 돌아간다.

④ 가상 명령어(pseudo op code)

Operand의 종류

리스트 9.1의 14~22번행에서 보듯이 Operand로는 레이블의 상위 바이트나 하위 바이트(DOS Tool kit 어셈블러의 것과 정반대이다.) 16진수, 10진수 문자(DOS Tool kit 어셈블러에는 MSB OM, MSBOFF 와 같은 명령을 썼지만 Merlin-Pro는 따옴표가 "인지 ' 인지로 구분한다), 2진수, 레이블 등이 사용된다. 이밖에 8진수도 사용할 수 있다.

```
                  1    **********************************
                  2    *                                *
                  3    *      Sample Programs 1         *
                  4    *                                *
                  5    **********************************
                  6
                  7              ORG   $1000      ;Assemble from here
                  8    Start     =     $500
                  9    Fi        =     $45
                 10
                 11    **** Date Type ****
                 12
1000: A9 00      13              LDA   #<Start    ;Lo Byte
1002: A9 05      14              LDA   #>Start    ;Hi Byte
1004: A9 90      15              LDA   #$90       ;Hex
1006: A9 5A      16              LDA   #90        ;Dec
1008: A9 C1      17              LDA   #"A"       ;Msb On
100A: A9 41      18              LDA   #'A'       ;Msb Off
100C: A5 AA      19              LDA   %10101010  ;Binary
100E: A5 45      20              LDA   Fi         ;1 Byte Label
1010: 8D 00 05   21              STA   Start      ;2 Byte Label
                 22
                 23    **** Date Define Opcode ****
                 24
1013: C1 C2 C3   25              ASC   "ABC"      ;Nomal Ascii Value
1016: 41 42 43   26              ASC   'ABC'
```

```
1019: 01 02 03  27          INV   "ABC"         ;Inverse Ascii Value
101C: 41 42 43  28          FLS   "ABC"         ;Flash Ascii Value
101F: C1 C2 43  29          DCI   "ABC"          ;Token Msb
1022: 41 42 C3  30          DCI   'ABC'         ;Token Lsb
1025: 00 00 00  31          DS    $10           ;Reserve 10 Byte
1028: 00 00 00 00 00 00 00 00
1030: 00 00 00 00 00
1035: 0C ED 45  32          DFB   12,$ED,Fi     ;1 Byte :Dex,Hex,Label
1038: 23 34 56  33          HEX   23,34,56      ;Hex Data
                34          DS    $500-*        ;Reserve Space To $500
0500: 0A 05     35          DA    Start+10      ;Two Byte Lo,Hi
0502: 05 0A     36          DDB   Start+10      ;Two Byte Hi,Lo
                37
                38  **** Extended Branch ****
                39
0504: B0 FA     40          BGE   Start         ;Equal To Bcs
0506: 90 F8     41          BLT   Start         ;Equal To Bcc

--End assembly, 62728 bytes, Errors: 0

Symbol table - alphabetical order:

   Fi      =$45        Start   =$0500


Symbol table - numerical order:

   Fi      =$45        Start   =$0500
```

리스트 9.1

Comment(주석문)

주석문을 2행의 첫번째 글자가 *일 경우로 어셈블할 때 무시하고 넘어간다. 그리고 Operand 다음의 탭부터도 주석문으로 사용된다.

Branch(분기)

BCS와 BCC는 우리들의 인식과는 조금 거리가 먼 느낌이 든다. 그래서 이 둘 대신 BGE(Branch Great Equal)와 BLT(Branch Less Than)를 사용하면 조금은 편해질 것이다. 물론 어셈블된 결과는 BCS와 BCC로 나타난다.

데이타 정의 Op Code

ORG : ORG는 프로그램을 어셈블할 때의 시작 번지를 나타낸다.

LST ON과 LST OFF

어셈블하면 리스트 9. 1과 같은 어셈블 리스트가 화면에 나오는데 소스의 양이 많으면 어셈블 리스트를 보는 데에도 많은 시간이 소요된다. 이때 소스의 첫머리에 LST OFF를 넣고 어셈블하면 어셈블 리스트가 나오지 않고 빠르게 어셈블된다. 이를 원래의 상태로 되돌리려면 LST ON을 소스에 넣으면 된다.

PUT

어셈블리 프로그램을 길게 작성할 경우 많이 사용되는 레이블들은 공통적이다. 그러므로 중복 사용

```
                    1    *********************************
                    2    *                               *
                    3    *     SAMPLE PROGRAMS 2         *
                    4    *                               *
                    5    *********************************
                    6
                    7              ORG   $300
                    8
                    9              PUT   SAMPLE2.1
                   >1    HOME      =     $FC58
                   >2    COUT      =     $FDED
                   10
0300: 20 58 FC     11              JSR   HOME
0303: A9 C1        12              LDA   #"A"
0305: 20 58 FC     13              JSR   HOME
0308: 60           14              RTS


--End assembly, 9 bytes, Errors: 0


Symbol table - alphabetical order:

?  COUT    =$FDED     HOME    =$FC58


Symbol table - numerical order:

   HOME    =$FC58  ?  COUT    =$FDED
```

리스트 9.2

되는 레이블을 Put 명령어로 처리하면 늘 쓰이는 내용을 저장시켜 놓고 Put을 사용하여 어셈블하면 편리하다.

DSK

DOS Tool kit의 어셈블러는 디스크에서 소스를 읽어서 어셈블하고 어셈블한 목적 코드를 디스크에 기록하기 때문에 메모리의 부족을 덜 느낀다. 이런 이점을 Merlin-Pro에서 가능하게 하는 Op Code가 DSK이다. 리스트 9. 3의 8번 행과 같이 DSK명령어와 파일 이름을 쓰고 어셈블하면 드라이브가 돌며 어셈블된 내용을 디스크에 기록한다.

```
                  1    **********************************
                  2    *                                *
                  3    *     SAMPLE PROGRAMS 3          *
                  4    *                                *
                  5    **********************************
                  6
                  7             ORG   $300
                  8             DSK   A
                  9    HOME     =     $FC58
0300: 20 58 FC   10             JSR   HOME
0303: A9 C1      11             LDA   #"A"
0305: 20 58 FC   12             JSR   HOME
0308: 60         13             RTS


--End assembly, 9 bytes, Errors: 0


Symbol table - alphabetical order:

   HOME    =$FC58


Symbol table - numerical order:

   HOME    =$FC58
```

리스트 9.3

SAV

여러 개의 프로그램을 동시에 어셈블할 때 사용되는 명령어가 SAV이다. 리스트 9. 4의 13번이나 20번 행과 같이 SAV와 파일 이름을 지정하고 어셈블하면 리스트 9. 5와 같이 13번 행에 이를 때

첫번째 프로그램을 디스크에 세이브하고 이후에 두번째 프로그램을 세이브한다. ORG가 같은 두 개 이상의 프로그램을 어셈블할 때에도 이용하면 편리하다.

⑤ MACRO

매크로는 펑션키와 비슷한 의미이다. 예를 들면 PRINT와 HTAB을 지정하고 프린트를 할 경우 펑션키가 없다면 PRINT, HTAB 등을 매번 써야 하지만 펑션키가 있다면 여러 번의 키동작을 한 번으로 줄일 수 있을 것이다.

어셈블러에서 어떤 문자를 프린트할 때 LDA#$A9와 JSR$FDED와 같이 매번 입력해야 한다. 그런데 이와 같은 것을 PRCHR$A9와 같이 쓸 수 있다면 아주 편리할 것이다. 앞의 예들은 간단한 것들이지만 반복해서 쓰는 내용이 길고 복잡한 것일수록 매크로의 효과는 매우 커질 것이다.

리스트 9. 6을 보면 매크로의 정의는 12번의 DO0과 25번의 FIN 사이에서 이루어진다. 여기서 PRCHR과 PRADRS는 매크로의 이름이고 MAC는 매크로임을 나타내는 Op Code, <<<는 그 매크로의 끝을 나타내는 Op Code이다.

이때]1,]2,]3과 같은 것은 메인 프로그램에서 전달되는 파라메터를 나타내는 변수와 같은 것이다.

```
**********************************
*                                *
*     SAMPLE PROGRAMS 4          *
*                                *
**********************************

         ORG   $300

         LDA   #"B"
         JSR   $FDED
         RTS

         SAV   B

         ORG   $300
         LDA   #"C"
         JSR   $FDED
         RTS

         SAV   C

```

리스트 9.4

35에서 41번 행까지의 프로그램은 어셈블리가 아니고 베이직같이 보일 정도이다.

Merlin—Pro의 디스크에 서브디렉토리인 LIB의 파일을 보면 유용한 매크로들이 있으니 참고하기

```
                 1    **********************************
                 2    *                                *
                 3    *     SAMPLE PROGRAMS 5          *
                 4    *                                *
                 5    **********************************
                 6
                 7              ORG   $300
                 8
0300: A9 C2      9              LDA   #"B"
0302: 20 ED FD  10              JSR   $FDED
0305: 60        11              RTS
                12
                13              SAV   B

Object saved as B,A$0300,L$0006,BIN

                14
                15              ORG   $300
0300: A9 C3     16              LDA   #"C"
0302: 20 ED FD  17              JSR   $FDED
0305: 60        18              RTS
                19
                20              SAV   C

Object saved as C,A$0300,L$0006,BIN

                21


--End assembly, 12 bytes, Errors: 0


Symbol table - alphabetical order:



Symbol table - numerical order:
```

리스트 9.5

```
********************************
*                              *
*      SAMPLE PROGRAMS 6       *
*                              *
********************************

PRINTAX  =     $F941
COUT     =     $FDED

**** Define Macro ****

         DO    0

PRCHR    MAC
         LDX   #]1
         JSR   COUT
         <<<

PRADRS   MAC
         LDX   ]1
         LDA   ]1+1
         JSR   PRINTAX
         <<<

         FIN

** Print Bload Address And Lenght **

BLEN     =     $AA60
BLADRS   =     $AA72
HOME     =     $FC58

         ORG   $300

         JSR   HOME
         >>>   PRCHR."A"
         >>>   PRCHR."$"
         >>>   PRADRS.BLADRS
         >>>   PRCHR.","
         >>>   PRCHR."L"
         >>>   PRCHR."$"
         >>>   PRADRS.BLEN
         RTS
```

리스트 9.6

바란다.

주의할 점으로 이 매크로 파일은 PUT 명령으로 자신의 프로그램에 포함시킬 때는 제대로 작동하지 않으므로 매크로 파일을 로드한 뒤 자신의 프로그램을 합치는 방법을 이용하면 된다.

⑥ 조건부 어셈블

리스트 9. 7을 보면 구조가 조금 이상한 것을 느낄 수 있다. 우선 ORG가 세 개이고 DO나 ELSE와 같은 Op Code들이 있다. 이것은 조건부 어셈블의 형태로 사용자로부터 어떤 숫자를 입력받아 여러 프로그램 중의 하나를 어셈블하는 것이다. 7번 행에 보면 Select란 레이블이 있고 KBD라는 Op Code가 있는데 이것을 어셈블하면 Select의 값이 얼마인지를 묻고 그 값을 입력받아서 9, 17, 24번과 같이 DO 다음의 값이 0이 되면 그 아래의 프로그램을 FIN을 만날 때까지 어셈블시킨다. 즉 입력한 값이

```
 1 *********************************
 2 *                               *
 3 *      SAMPLE PROGRAMS 7        *
 4 *                               *
 5 *********************************
 6
 7 Select   KBD              ;Input A Value
 8
 9          DO    Select-1
10          ELSE
11
12          ORG   $1000
13          LDA   #"A"
14
15          FIN
16
17          DO    Select-2
18          ELSE
19
20          ORG   $2000
21          LDA   #"B"
22
23          FIN
24          DO    Select-3
25          ELSE
26
27          ORG   $3000
28          LDA   #"C"
29
30          FIN
```

리스트 9.7

2이면 9번 행에서 Select-1 = -1이 되어 실행되지 않고 10번 행의 ELSE, 15번 행의 FIN을 거쳐서 17번 행의 Select-2 =0이 되므로 20번과 21번 행을 어셈블하고 23, 24, 25, 26번을 거쳐 어셈블이 끝난다.

§ 2. Z-BASIC

지금까지 애플 유저들은 MSX나 SPC 기종의 CIRCLE, PAINT 등의 그래픽 명령어와 다른 편리한 명령어를 부러워한 경험이 있을 것이다. 그러나 이 Z-BASIC을 접하고 나면 다른 기종의 BASIC 만큼, 아니 더 뛰어난 기능을 느낄 수 있을 것이며 애플Ⅱe에서는 더블 고해상도와 더블 저해상도도 쓸 수 있어 Ⅱe의 전용 BASIC이라 해도 과언이 아닐 것이다.

Z-BASIC의 기본 모드는 다음과 같다.

- 스크린 모드 : 15개
- TEXT 모드

  0 and 8 = 40×24 배열

  2 and 10 = 80×24 〃

  4 and 12 = 40×24 〃

  6 and 14 = 80×24 〃
- Graphic 모드

  1 and 9 = 40×48 저해상도 그래픽

  3 and 11 = 80×48 더블 저해상도 그래픽

  5 and 13 = 280×192 고해상도 그래픽

  7 and 15 = 560×192 더블 고해상도 그래픽

그리고 Z-BASIC에는 64K와 128K가 있는데 64K는 ProDOS의 램디스크를 이용하여 속도가 빠르지만 128K는 그렇지 못해서 컴파일 속도가 느리다. 그러므로 128K보다는 64K를 더 권장하고 싶다.

1. Z-BASIC의 입출력 명령어

APPEND line#-1, filename-1, line#-2, filename-2

애플의 TEXT File의 Append 명령과 비슷하다.

기능 : filename-1이라는 프로그램이 메모리에 있을 때 filename -2라는 프로그램을 line#-1에서 filename-2의 line#-2까지 로드한다.

[L] DIR [F] [pathname]

L : 프린터로 출력

F : 파일 이름

기능 : 디렉토리 파일을 보내준다. 애플의 Catalog 명령과 비슷하다.

CLS

화면을 클리어시킨다.

CONFIG

Z-BASIC 상태를 세트시킨다.

EDITOR or [ESC] key

풀 스크린 에디트 모드(full screen edit mode)로 들어간다.

FIND [*]

* : 숫자 or 문자 (여러 문자도 가능)

특정 문자를 찾는다.

HELP [NUM]

num : 숫자

Help 모드로 들어간다.

[L] LIST

프로그램을 본다(LLIST : 프로그램을 프린터로 출력시킨다).

LOAD [*]

* : 파일 이름

프로그램을 보조 메모리에서 메인 메모리로 옮긴다.

MEM

메모리 사용 영역을 보여준다.

MERGE [*] pathname

* : 파일 이름

메모리에 있는 프로그램과 디스크의 프로그램을 합친다(디스크의 파일은 TXT파일 형식이어야만 된다).

NEW

메모리에 있는 프로그램을 지운다.

ONLINE

프로도스 세트 상황을 보여준다.

예 ONLINE

S3, D2 = /RAM

S6, D1 = /ZBASIC. 64K

PATH *

서브디렉토리로 패치(patch)시킨다(파일 형태가 DIR의 형태여야 한다).

예 PATH SAMPLE, PROGRAMS를 한 후 DIR해 보라.

디렉토리 이름이 /ZBASIC. 64K/SAMPLE. PROGRAMS로 되어 있을 것이다.

여기서 다시 ZBASIC. 64K의 디렉토리로 돌아가려면 PATH ZBASIC. 64K해주면 된다.

QUIT

ZBASIC에서 빠져나와 다른 시스템 파일을 Run시킬 수 있다.

RENAME filename 1, filename 2

파일 이름을 바꾼다.

RENUM [NEW], [old], [inc]

[NEW] : 바뀔 행번호

[old] : 바꾸기 전의 행번호

[inc] : 증가량

모든 것을 생략하고 그냥 RENUM을 하면 처음부터 10단위로 증가한다.

RUN *

프로그램을 실행시킨다.

SAVE [* or +] filename

*, +는 파일 이름이 아님.

프로그램을 디스크에 저장.

(파일 이름 앞에 "*"를 붙이면 프로그램이 ASCII Code로 저장된다. "+"를 붙이면 행번호를 없애고 ASCII Code로 저장된다.)

RIGHTS(str$, len)

str$: 문자열

len : 0~255 사이값

기능 : 문자열 str$에서 len개만큼 문자를 취한다.

SPACES(expr)

expr : 0~255 사이의 수

기능 : expr개의 공간 문자열을 만든다.

예제 "COLOR"를 " C O L O R"로 프린트하는 프로그램을 만들어보자.

```
10 N$ = "COLOR"
```

```
20 FOR I = 1 TO LEN(N$)
30 PRINT MID$(N$, I, 1); SPACE$(I);
40 NEXT I
RUN
C O   L    O       R
ZBASIC Ready
```

SPC(expr) [;]

expr : 0~255 사이의 정수 또는 수식.

기능 : expr 개의 공백을 프린트한다.

* 반드시 PRINT문과 함께 사용되어야 한다.

STRS(expr)

expr : 수치

기능 : 수치 expr을 문자열로 변환한다.

VAL(str$)

str$: 문자열

기능 : 문자열을 수치로 변환한다.

TIME$

기능 : 시간, 분, 초를 저장하고 있다.

```
PRINT TIME$
00 : 00 : 00
```

UNS$(expr)

expr : 0~65535 사이의 수치

기능 : 수치의 앞자리부터 다섯 자리까지 0을 넣어 문자열로 바꾼다.

예제 예제를 보면 쉽게 이해할 수 있다.

```
10 A$ = UNS$(10)
20 B$ = UNS$(65535)
30 PRINT A$, B$
RUN
00010     65535
```

STRING$[(expr, expr1) or (expr, strS)]

expr, expr1 : 0~255 사이의 값

str$: 문자열

기능 : ① ASCII Code expr1에 해당하는 문자를 expr 개의 문자열로 바꾼다.
② str$의 첫번째 글자를 expr개의 문자열로 바꾼다.

주 str$가 공문자열(" "의 상태)이면 에러가 발생한다.

예제
```
10 A$ = "APPLE"
20 X$ = STRING$(5, A$)
30 Y$ = STRING$(5, 66) ; REM 66→B
40 PRINT X$, Y$
RUN
AAAAA     BBBBB
```

ASC(str$)

str$: 문자

기능 : str$의 문자를 ASCII Code 값으로 바꾼다.

예제
```
10 A$ = "A"
20  X = ASC(A$)
30  PRINT X
RUN
65
```

BIN$(expr)

expr : -65535~65535 사이의 수치

기능 : 수치를 16bit의 2진수의 문자열로 만든다.

예제
```
10  X = 10
20  A$ = BIN$(65535)
30  B$ = BIN$(X)
40  C$ = BIN$(X +246)
50  PRINT A$
60  PRINT B$
70  PRINT C$
RUN
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
```

CHR$(expr)

expr : 0~255 사이의 수치

기능 : ASCII Code expr을 이에 해당하는 문자로 바꾼다.

예제
```
10 PRINT CHR$(65)
RUN
A
```

DATE$

기능 : 시스템 내에 있는 날짜를 표시한다

HEX$(expr)

expr : −32168~+65535 사이의 수치

기능 : 십진수 expr을 16진수의 값을 나타내는 문자열로 표현한다.

예제 임의의 수를 입력하여 16진수로 바꾸는 프로그램을 만들어 보자.
```
10  INPUT A
20  X$ = HEX$(A)
30  PRINT A;"−−〉 &  H" ; X$
RUN
?44
44−−〉 & H2C
```

INKEY$

기능 : 키보드에서 문자 하나를 받아들인다. 애플에서 PEEK(-16384)와 거의 같다.

예제
```
10 X$ = INKEY $
20  PRINT X$
30  IF X$ = "A" THEN END : ELSE
    GOTO 10
```

NSTR$(pos, str$1, str$2)

Pos : 1~255 사이의 수치

str$1, str$2 : 문자형 변수

기능 : 문자열 str$1에 문자열 str$2가 있는지 조사한다. Pos는 문자열 str$1의 처음부터 Pos번째 부터 찾는다.

예제
```
10  A$ = "STAR"
20  B$ = "T"
30  PRINT INSTR$(1, A$, B$)
40  PRINT INSTR$(3, A$, B$)
```

```
RUN
2
0
```

LEN(str$)

str$: 문자열

기능 : 문자열 str$의 길이의 숫자로 취한다. *, 공백도 포함한다.

LEFT$(str$, len)

str$: 문자열

len : 0~255 사이의 수치

기능 : 문자열의 왼쪽부터 len개를 취한다.

예제
```
10 A$ = "Apple //e"
20 FOR I = 1 TO LEN(CA$)
30 PRINT LEFT$(A$, I)
40 NEXT
RUN
A
Ap
App
Appl
Apple
Apple/
Apple//
Apple//e
```

MID$(str$, pos, len)

str$: 문자열

pos : 1~255 사이의 정수

len : 0~255 사이의 정수

기능 : 문자열 str$의 pos번째의 len개의 문자를 빼낸다.

① X$ = MID$(str$, pos, len)

문자열 str$의 pos번째의 len개의 문자를 X$에 담는다.

② MID$ (str$, pos, len) = X$

문자열 X$를 문자열 str$의 pos번째 len개의 문자에 담는다.

예제
```
10 A$ = "LOVE IS BLUE"
20 B$ = "SAD"
30 PRINT A$
40 MID$(A$, 9, 2) = B$
50 PRINT A$
60 MID$(A$, 9, 4) = B$
70 PRINT A$
RUN
LOVE IS BLUE
LOVE IS SAUE
LOVE IS SAD
```

OCT$(expr)

expr : -32768～+65535의 수치

기능 : 십진수 expr의 값을 8진수로 나타낸다.

* expr이 음수이면 2의 보수 형태를 취한다.

OCT$(-X)는 OCT$(65535-X)와 같다.

예제 10～18 사이의 수를 8진수로 변환하는 프로그램을 만들어 보자.

```
10 FOR I = 10 TO 18
20 PRINT I ; "-->& O" ; OCT$(I)
30 NEXT I
RUN
10 --> & 012
11 --> & 013
12 --> & 014
13 --> & 015
14 --> & 016
15 --> & 017
16 --> & 020
17 --> & 021
18 --> & 022
```

MKI$(expr)

expr : 정수

기능 : 정수 expr을 2진 코드에 해당하는 문자열로 바꾼다.

예제 C자와 Ap를 프린트하자.

```
10 A = 67
20 B = 28737
30 PRINT MKI$(A), MKI$(B)
RUN
C
```

CVI(str$)

str$: 문자열

기능 : MKI$와는 상반된 기능으로 문자열을 수식으로 변환한다.

예제
```
10 A$ = "C"
20 B$ = "Apple//e"
30 Print A$, CVI(A$)
40 PRINT B$, CVI(B$)
RUN
C          67
Apple//e 28737
```

2. 그래픽 명령어

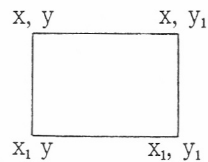
그래픽 화면은 가로 1024, 세로 768까지 지정해 줄 수 있다.

BOX

형식 : BOX x, y TO x_1 y_1

x y x_1 y_1 : 수치

기능 : (x, y)에서 (x_1, y_1)의 대각선을 중심으로 BOX를 그린다.



형식 : BOXFILL x, y TO x_1, y_1

기능 : 박스를 그리고 박스 안을 색칠한다.

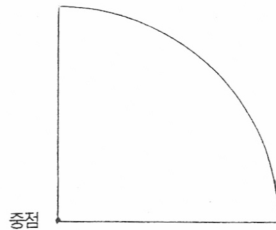
CIRCLE

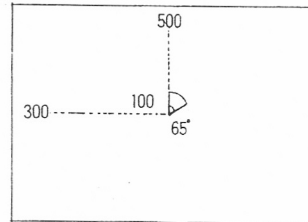
형식 : CIRCLE x, y, r[TO s, n]

x, y, r : 수치 또는 수식

기능 : x, y를 기준으로 반지름 r로 그린다. s(시작점 각도), n(끝점 각도)에서 원의 중심까지 선을 그린다.

예제 CIRCLE 500, 300, 100, TO 0, 65

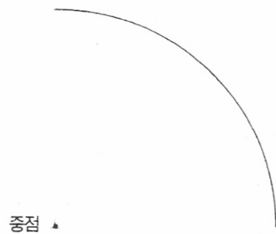


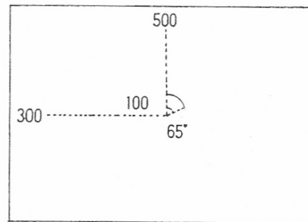


형식 : CIRCLE x, y, r[PLOT s, n

기능 : 기능은 위와 비슷하나 중심까지 선을 그리지 않는다.

예제 CIRCLE 500, 300, 100 PLOT 0, 65

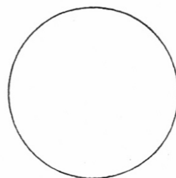


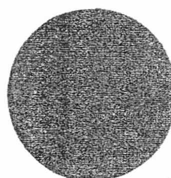


형식 : CIRCLEFILL x, y, r

기능 : 원을 그리고 원 안을 칠한다.

예제
```
10 MODE 7 : CLS
20 CIRCLE 200, 300, 100
30 CIRCLEFILL 500, 300, 100
RUN
```





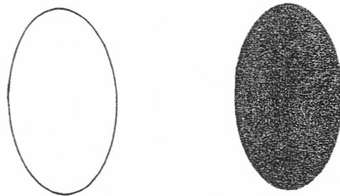
RATIO width, height

width : 원의 가로 반지름

height : 원의 세로 반지름

기능 : 원의 가로 반지름과 세로 반지름을 받아서 타원 등으로 만든다.

예제
```
10 MODE 7 : CLS
20 RATIO 100, 200
30 CIRCLE 500, 300, 100
40 CIRCLEFILL 500, 300, 100
RUN
```



COLOR

형식 : COLOR = n

n : -1~15 사이의 정수

기능 : 그래픽의 색을 정한다.

*0 = 검은색, -1 = 흰색, 1~15 = 나머지 색

FILL(x, y)

기능 : x, y를 기준으로 색을 칠한다.

* MSX, SPC 등의 PAINT와 같은 기능의 명령어이다.

PRINT

형식 : PRINT

기능 : 현재의 모드에서 PRINT의 기능

형식 : PRINT@(x, y)

x, y : 수치

기능 : 텍스트 모드에서 가로x, 세로 y축에 위와 같은 기능으로 프린트된다.

형식 : PRINT %(x, y)

x, y : 수치

기능 : 그래픽 모드에서 가로 x, 세로 y축에 위와 같은 기능으로 프린트된다.

INPUT

형식 : INPUT

기능 : 현재의 모드에서 INPUT의 기능

형식 : INPUT@(x, y)

x, y : 수치

기능 : 텍스트 모드에서 가로x, 세로 y축에 위의 INPUT과 같은 기능으로 작동한다.

형식 : INPUT%(x, y)

x, y : 수치

기능 : 그래픽 모드에서 가로x, 세로 y축에 위와 같은 기능으로 동작한다.

POINT(x, y)

x, y : 수치

기능 : x, y축에 있는 도트의 On, Off 상태를 읽는다. 타기종의 POINT명령과 유사하다.

* on일 때 값>0, off일 때 값 = 0

예제
```
10 MODE 7 : CLS
20 COLOR = 3 : PLOT 100, 100
30 COLOR = O : PLOT 400, 100
40 A = POINT (100, 100)
50 B = POINT (400, 100)
60 PRINT%(100, 150) "DOT =" A
70 PRINT%(400, 150) "DOT =" B
RUN
•
DOT = 2          DOT = 0
```

CLS, CLS PAGE

기능 : 화면 또는 PAGE를 클리어시킨다.

PLOT

형식 : PLOT x, y

기능 : x, y축에 점을 찍는다.

형식 : PLOT x, y TO x_1, y_1

기능 : x, y에서 x_1, y_1으로 선을 긋는다.

* 애플의 PLOT과 기능 및 사용 방법이 같다.

LOCATE x, y,[, C]

x, y : 수치
C : C = 0이면 커서가 나타나지 않는다.
C = -1이면 커서가 나타난다.

MODE = n
n : 0~15 사이의 수치
기능 : n에 해당하는 스크린으로 간다.

3. DISK 파일 입출력 명령어

ERROR
기능 : Error의 번호를 담고 있다.

ERRMSGS(err)
err : 에러 넘버
기능 : 에러 넘버에 대한 에러 메시지를 보여 준다.

KILL filename
filename : 파일 이름
기능 : 디스크 내의 파일을 지운다.

ON ERROR GOSUB line#
line# : 라인 넘버
기능 : DISK 작동시 에러가 생겼을 때 line#로 점프한다.

ON ERROR RETURN
기능 : DISK 작동시 에러가 생기면 Z -BASIC으로 나간다.

OPEN "ftype", filenumber, filename$ (optional RECORD length)
ftype : "R" 읽기/쓰기를 할 수 있다(랜덤).
"O" 쓰기만 할 수 있다(순차적).
"I" 읽기만 할 수 있다(순차적).
filenumber : 파일 번호(1~15사이의 정수식)
filename$: 파일명
optional RECORD length : 레코드의 길이 설정(랜덤 파일일 때)
기능 : 파일을 오픈시킨다.

CLOSE# fnum1, ……, fnum∞
fnum : 파일 번호

기능 : 디스크 파일의 입출력을 종료한다.

* fnum만 종료

CLOSE#

기능 : 모든 fnum를 종료.

PRINT# filenumber, data, ……

filenumber : 파일 번호

data : 디스크에 넣을 수치 또는 스트링

기능 : 데이타를 디스크에 ASCII Code로 기록한다.

WRITE# fnum, var, var$; len,……

funm : 파일 번호

len : var$를 사용, 즉 스트링을 쓸 때는 레코드 길이를 설정한다.

기능 : 파일에 데이타를 기록한다.

* PRINT 보다 빠르다.

INPUT# fnum, var, var

fnum : 파일 번호

var : 데이타

기능 : 데이타를 디스크에서 ASCII Code로 읽어들인다.

LINEINPUT# fnum, var$

fnum : 파일 번호

var$: 문자열 변수

기능 : 데이타를 디스크에서 구분없이 한 행 전체를 입력하여 문자열 변수에 할당한다.

READ# fnum, var, var$; len

fnum : 파일 번호

var, var$: 수치, 문자열 변수

len : var$를 사용, 즉 문자열을 쓸 때는 레코드 길이를 설정한다.

기능 : 파일의 데이타를 읽어 변수에 할당한다.

* INPUT#보다 빠르다.

RENAME fn1, fn2

fn1, fn2 = 파일 이름

기능 : 파일 fn1을 fn2로 바꾼다.

LOC(fnum)

fnum : 파일 번호

형식 : X = LOC(fnum)

기능 : 파일의 현재 위치를 알려준다.

LOF(fnum)

fnum : 파일 번호

기능 : 파일에서 맨 마지막 위치를 알려준다.

CALL nnnnn

nnnnn : 0～65535 사이의 수치

기능 : 기계어 서브루틴을 직접 호출한다.

CALL LINE n

n : 수치 또는 테이블

기능 : MACHLG 루틴을 호출하는 데 사용

DEFUSR n = address

n : 0～9 사이의 정수

adress : 번지

기능 : 사용자가 USR함수에 의해 호출할 기계어 서브루틴을 정의하는 데 사용한다.
또 MACHLG 루틴을 호출하는 데도 사용된다.

USRn(expr)

n : 0～9 사이의 수치

expr : 데이타 인수

기능 : 인수 n을 기계어 서브루틴에 건네주고 서브루틴을 호출한다.

PEEK(address)

address : 읽을 주소(기계어 번지)

기능 : 기계어 번지 address에서 한 바이트 읽어낸다.

PEEKWORD(addr)

addr : 읽어낼 기계어 번지

기능 : 읽어낼 번지와 그 다음 번지를 읽어낸다.

예제 ＊ 60 + 10 ×256 = 2620

```
10 POKE  &300, 60
20 POKE  &301, 10
30 X = PEEKWORD(&300)
40 PRINT X
RUN
```

2620

POKE addr, byte

byte : 0~255 사이의 수치

addr : 쓸 기계어 번지

기능 : 번지 addr에 수치인 byte를 넣는다. 이 때에는 한 byte가 들어간다.

POKEWORD addr, expr

addr : 쓸 기계어 번지

expr : 0~65535 사이의 수치

기능 : 번지인 addr과 addr의 다음 번지에 expr을 넣는다.

MACHLG var, var, ……

var : 수치

기능 : 이것은 Z-BASIC의 특수한 명령어 중 하나이다. var에 기계어 데이타를 가지면서 이것을 기계어 번지에 Poke시키지 않고 실행시킬 수 있다.

예제 화면 상단에 Apple//e를 프린트하자.

```
10 CALL LINE "APPLE"
20 END
30 "APPLE" MACHLG  &20, &60, &FB, &60 REM JSR $FB60 ; FB60→ Apple//
   e를 프린트시키는 번지 RTS
RUN
Apple//e
```

*MACHLG 루틴은 주로 CALL LINE이나 DEFUSR에 의해 불려진다.

DELAY expr

expr : 수치

기능 : 시스템을 expr/1000 시간만큼 지연시킨다.

LINEINPUT

기능 : INPUT과 같은 기능이나 커머(,)나 그밖의 것도 모두 문자열로 받아들인다.

LPRINT

기능 : 프린터로 출력시킨다.

POS(n)

n : 수치

기능 : 현재의 커서의 위치 표시. n = 0 이면 스크린, n = 1 이면 프린터, n = 2 이면 디스크

TAB(expr)

expr : 0~255 사이의 수치

기능 : 화면에 지정된 인쇄 위치 expr로 이동한다.

SPC(expr)

expr : 0~32767 사이의 수치

기능 : expr개의 공백을 프린트한다.

MOUSE(n)

n : 0~3 사이의 수치

기능 : n = 0 마우스 초기화

n = 1 마우스 X축 읽기

n = 2 마우스 Y축 읽기

n = 3 마우스 버튼 감지

PRINT USING str$; var

str$: 특별한 형식의 문자로 구성된 문자열

var : 수식 목록

기능 : 특정한 형태로 문자열이나 숫자를 프린트한다.

예제 PRINT USING "###, ###. ##" ; 1223. 125 = 1, 223.13

PRINT USING "$##, ###. ##" ; 999.3212 = $999.32

PRINT USING "*##, ###. ##" ; 32. 598 = ****32.60

PRINT USING "+##, ###, ##" ; 32.598 = +32.60

PRINT USING "+##, ###, ##" ; -45.199 = -45.20

PRINT USING "##/##/#" ; 33185 = 03/31/85

PRINT USING "##:##:##" ; 12349 = 1 : 23 : 49

PRINT USING ".#######" ; 1.2E-5 = . 000012

PRINT USING "%###. ##"; 123.355 = % 123.36

PRINT USING "@###, ###" ; 34.349 = @ , 34

A$ = "##": PRINT USING A$: 89. 9 = 90

WIDTH expr

expr : 0~80 사이의 수치

기능 : 스크린의 가로 칸수를 세트한다.

WIDTH LPRINT expr

expr : 0~80 사이의 수치

기능 : 프린터의 가로 칸수를 세트한다.

SOUND freq, duration

freq : 높이

duration : 길이

기능 : 애플 스피커로 Freq와 duration에 해당하는 소리를 출력한다.

CLEAR

기능 : 모든 변수를 지워버린다.

DEF┌INT ┐ , var - var
│SNG│
│DBL│
└STR┘

var -var : 시작 문자— 끝 문자

기능 : 변수를 정수, 단정도 수, 배정도 수, 문자열 등으로 설정한다.

DEF FN name(var [, var …]) = exr

name : 함수명

var : 인수(정의식에서 사용되는 변수명으로 함수가 호출될 때 값으로 대치된다.)

exr : 수식(함수의 기능을 수행)

기능 : 프로그램에서 사용자가 독자적으로 함수를 정의한다.

DIM var (len〉 [, var(n, n)] ……)

var : 변수

len, n : 첨자

기능 : 배열변수의 첨자와 할당된 기억 장소의 최대값을 지정한다.

IF X THEN A ELSE B

A, B : 수식

기능 : 논리식이 참인지 거짓인지를 판단해서 지정한 행번호나 문으로 실행을 옮긴다.

END

기능 : 프로그램 수행을 끝낸다.

LET var = expr

var : 변수

expr : 수식

기능 : 수식을 계산하여 var변수에 할당한다.

FOR var = X1 TO X2 STEP Y

var : 변수

X1, X2 : 수치나 수식

Y : 증감량

기능 : NEXT와 함께 쓰여서 X1에서 X2만큼 루프를 돈다.

NEXT var

var : 변수

기능 : FOR 루프가 끝날 때까지 FOR로 돌아간다.

GOTO expr

expr : 라인이나 레이블

기능 : 라인이나 레이블로 점프해 간다.

GOSUB expr

expr : 라인이나 레이블

기능 : 서브루틴으로 점프해 간다.

RETURN line

line : 행번호

기능 : 뒤에 Line을 정해주지 않으면 GOSUB한 곳으로 돌아간다.
행번호를 정해주면 그곳으로 간다.

STOP

기능 : 실행을 멈춘다.

LONG FN name(var)

name : 이름

var : 변수

기능 : DEF FN과 비슷하다.
함수를 사용할 수도 있으나 주로 서브루틴으로 이용한다.

END FN

기능 : LONG FN과 함께 쓰여서 LONG FN이 끝나고 FN으로 오기 전의 루틴으로 돌아간다.

예제 I에 10을 넣고 LONG FN 루틴을 써서 그값에 1을 더하는 프로그램을 만들어 보자.

```
10 LONG FN A(I)
20 PRINT "START FN" ;
30 I = I + 1
40 END FN
50 I = 10
60 B = FN A(I)
```

```
70 PRINT B
80 PRINT "END"
RUN
START FN 11
END
```

＊ 실행 순서

10→50(I = 10) →60→10→20→30(I = I + 1) →40→60→70→80

실행 순서에서 보듯이 처음에는 LONG FN 루틴을 실행시키지 않고 FN A(I)를 만난 후에 LONG FN을 실행시켰다.

LONG IF cond XELSE

cond : 수식

기능 : 이것도 위에서처럼 IF문과 상당히 비슷하나 IF문보다 더 긴 처리를 할 수 있다. END IF 문과 같이 쓰인다.

예제 A와 B를 입력하여 대소를 비교하자.

```
10 INPUT "A, B = ?" ; A, B
20 LONG IF A < B
30     PRINT " . . . . A < B . . . "
40 XELSE
50     LONG IF A > B
60       PRINT " . . . . A > B . . ."
70     XELSE
80       LONG IF A = B
90           PRINT ". . . .A = B . . ."
100      END IF
110    END IF
120 END IF
RUN
A, B = ? 10, 20
. . . . A < B . . .
RUN
A , B = ? 20, 10
. . . .A > B . . .
RUN
```

. . . . A = B . . .

ON expr GOSUB a, a1 a2, …

expr : 식

a : 행번호

기능 : 식의 값에 따라 a, a1, a2, … 등으로 GOSUB한다.

ON expr GOTO a, a1, a2, …

expr : 식

a : 행번호

기능 : 식의 값에 따라 a., a1, a2, … 등으로 GOTO한다.

PST$ (vy.)

vy. : 변수

기능 : 내장되어 있는 문자열 변수이다.

READ PSTR$(vy.)

vy. : 변수

기능 : 데이타를 PSTR$에 넣는다.

SWAP X, Y

X, Y : 변수

기능 : X변수의 내용을 Y에 담고 Y에 있던 내용을 X에 담는다.

예제
```
10 X = 10 : Y = 20
20 PRINT X, Y
30 SWAP X, Y
40 PRINT X, Y
RUN
10        20
20        10
```

TRON

기능 : 이 명령어는 Apple의 TRACE와 같은 명령어이다. 화면상에 지금 실행되고 있는 행번호를 보여준다.

TRONB

기능 : 이것은 얼핏 보면 TRON과 비슷한 것 같으나 그렇지 않다. 어떤 프로그램을 실행시키면 Reset을 제외하고는 프로그램이 끝날 때까지 멈추지 않는다. 그 때 CTRL－C키로 멈출

수 있게 해준다.

TRONS

기능 : 프로그램 수행을 중지시키고 중지시킨 것을 계속 수행시키는 것이다. 중지시킬 때나 계속 수행시킬 때 CTRL－Z로 조정한다.

TRONX

기능 : TRONB와 같은 기능이나 다른 점은 TRONB는 프로그램 실행 도중 언제든지 실행되지만 TRONX는 TRONX라는 명령어를 만날 때에만 작동된다는 것이다.

TROFF

기능 : 위의 TRON, TRONB, TRONS, TRONX를 모두 취소한다.

DO～UNTIL

기능 : 논리식이 참이 될 때까지 루프를 돈다.

형식 : DO

.

.

문

.

.

UNTIL 논리식

예제 어떤 수를 입력하여 그 수에서 1씩 빼 0이 될 때까지 프린트하자.

```
10 INTUT "X = " ; X
20 DO
30 PRINT X
40 X = X - 1
50 UNTIL X < 1
60 PRINT "FINISH"
```

WHILE～WEND

기능 : 조건이 참인 동안 루프를 돈다.

형식 : WHILE 논리식

≀

문

≀

WEND

예제 어떤 수를 입력하여 0이 될 때까지 프린트하자.

```
10 INPUT "X = " ; X
20 WHILE X > = 1
30 PRINT X
40 X = X - 1
50 WEND
60 PRINT "FINISH"
```

VARPTR

기능 : 메인 메모리에 있는 수치형 변수의 위치를 얻는다. 이것은 주로 변수의 주소를 알아 기계어 서브루틴에 넘겨 줄 때 사용된다.

예제

```
10 X = 10
20 Y = VARPTR(X)
30 PRINT Y, HEX$(B)
RUN
-31750 83FA
```

*앞의 결과는 프로그램에 따라 바뀔 수 있다.

아래 내용은 어떤 기종의 BASIC에서도 실행되는 공통 명령어이므로 생략하기로 한다.

+, -, ∧, *, /, MOD

ABS	ASC	ATN	COS	EXP	FIX	FRAC	INT
LOG	RND	SGN	SIN	SQR	TAN	VAL	

이상으로 Z-BASIC의 명령어 설명을 마친다. 부족하나마 여러분에게 도움이 되길 빌며 다음의 예제 프로그램으로 이해에 좀 더 도움이 되었으면 한다.(Z-BASIC은 Mouse Text도 모드 5와 7에서 지원해 줌을 밝힌다)

```
00010 MODE 7:CLS
00020 FOR I = 50 TO 767 STEP 100
00030   PLOT 0,I TO 1023,I
00040 NEXT
00050 FOR I = 10 TO 950 STEP 67
00060   BOX FILL I,767 TO I+50,768-RND(768)
00070 NEXT
00080 PRINT @(35,0)"BAR GRAPH";
00090*DO :UNTIL LEN(INKEY$)
00100 CLS
00110 FOR I = 50 TO 1023 STEP 50
00120   PLOT I,0 TO I,767
00130 NEXT I
00140 FOR I = 50 TO 767 STEP 50
00150   PLOT 0,I TO 1023,I
00160 NEXT
00170 PLOT 10,767
00180 FOR I = 10 TO 1023 STEP 100
00190   J = RND(768)
00200   PLOT TO I,J
00210   CIRCLE FILL I,J,5
00220 NEXT
00230 PRINT @(35,0)"LINE GRAPH";
00240*DO:UNTIL LEN(INKEY$)
```

리스트 9. 8 그래픽 예제

```
00010 LONG FN Prefix$(Path$)
00011   REM This sets up the ProDOS parameter block
00012   POKE &1F00,1: POKE WORD &1F01,VARPTR(Path$)
00013   REM Check for a string to set the prefix to
00014   x = LEN(Path$)
00015   REM If no string, then just return the current prefix
00016   IF x = 0 THEN "set_done"
00017   MACHLG &A9, &C6, &20, &0803
00018   "set_done" MACHLG &A9, &C7, &20, &0803
00019 END FN = Path$
00100 REM This program is intended to show you how to use the Prefix$ function
00110 REM To include the function in your programs, type
00120 REM "APPEND 10 PREFIX.FN"
00130 REM Then use ZBasic statements as in the following lines of program
```

```
00140 REM This is just about what the Editor does with the PATH command
00150 REM

00160 A$ = FN Prefix$(""): REM This gets the current prefix into A$
00170 PRINT "Current prefix is "A$
00180 INPUT "Enter new prefix -> ";A$
00190 A$ = FN Prefix$(A$): REM This will set the new prefix to A$
00200 PRINT "New prefix is "A$
00210 END
```

리스트 9. 9 PATH 프로그램

```
00010 FOR Mode = 1 TO 7 STEP 2
00020   MODE Mode : CLS
00030   GOSUB "DRAW PALLETTE"
00040 NEXT Mode
00050 Mode = 1
00060 DO
00070   MODE Mode
00080   Mode = Mode + 2 : IF Mode > 7 THEN Mode = 1
00090 DO : A$ = INKEY$ : UNTIL LEN(A$)
00100 UNTIL A$ = "S" OR A$ = "s"
00110 MODE 7 : LOCATE 0,18 : END
00120 "DRAW PALLETTE"
00130 FOR I = 1 TO 15
00140   COLOR = I
00150   BOX FILL 0,I*32 TO 1023,I*32+32
00160 NEXT I
00170*DO : UNTIL LEN(INKEY$)
00180 RETURN
```

리스트 9. 10 그래픽 모드에 따른 칼라 비교

```
00010 REM ANY ARRAY TYPE MAY BE USED
00020 REM IF YOU WANT SA(500) TO BE A STRING ARRAY, USE DEFSTR S ON THIS LINE
00030 REM IF STRINGS ARE USED, SET THE LENGTH OF SA(500) WITH DEFLEN=30 OR WHATE
VER
00040 DIM SA(500)
00050 NI = 500:REM CHANGE DIM 500 AND THIS 500 IF ARRAY SORT LARGER
00060 FOR X = 0 TO NI
00070   SA(X)=RND(1000):REM STORES RANDOM NUMBERS FOR SORTING
00080   REM:SA(X)=STRING$(RND(20),RND(60)+32) ONLY IF SA() IS A STRING ARRAY
00090 NEXT
```

```
00100 PRINT "START TIME: ";TIME$
00110 GOSUB "QUICK SORT"  :REM OR SHELL SORT
00120 PRINT "FINISH TIME: ";TIME$
00130 FOR X=NI-10 TO NI
00140   PRINT SA(X):REM PRINT LAST 10 ELEMENTS TO MAKE SURE SORT WORKED
00150 NEXT
00160 END
00200 "QUICK SORT"
00201 DIM ST(30,1) : REM MOVE THIS TO THE BEGINNING OF YOUR PROGRAM
00202 SP = 0 : ST(0,0) = 0
00203 SF = -1 : ST(0,1) = NI
00204 DO
00205   L = ST(SP,0) : R = ST(SP,1) : SP = SP - 1
00206   DO
00207     LI = L : RI = R : SA = SA((L + R) / 2)
00208     DO
00209       WHILE SA(LI) < SA
00210         LI = LI + 1
00211       WEND
00212       WHILE SA(RI) > SA
00213         RI = RI - 1
00214       WEND
00215       LONG IF LI <= RI
00216         SWAP SA(LI), SA(RI)
00217         LI = LI + 1 : RI = RI - 1
00218       END IF
00219     UNTIL LI > RI
00220     LONG IF (R - LI) > (RI - L)
00221       LONG IF L < RI
00222         SP = SP + 1 : ST(SP,0) = L : ST(SP,1) = RI
00223       END IF
00224       L = LI
00225     XELSE
00226       LONG IF LI < R
00227         SP = SP + 1 : ST(SP,0) = LI : ST(SP,1) = R
00228       END IF
00229       R = RI
00230     END IF
00231   UNTIL R <= L
00232 UNTIL SP = -1
00233 RETURN : REM QUICKSORT FINISHED HERE
00300 "SHELL SORT" Y=NI
00301 "Z1" Y = Y/2
00302 IF Y = 0 THEN RETURN: REM SORT COMPLETE
```

```
00303 Z99=NI-Y
00304 FOR K9 = 1 TO Z99
00305   I = K9
00306   "X2" E2 = I+Y
00307   REM: IN LINE BELOW CHANGE <= TO >= FOR DESCENDING ORDER
00308   IF SA(I) <= SA(E2) THEN "X3" ELSE SWAP SA(I),SA(E2)
00309   I=I-Y
00310   IF I>0 THEN "X2"
00311 "X3" NEXT K9
00312 GOTO "Z1"
00313 END
```

리스트 9. 11 소트의 예제

```
00005 REM >>>>>>>>>> 'HOUSE' DRAWS A 3'D HOUSE <<<<<<<<<<<<<<
00010 DEFDBL A-Z : DEFINT I,J,K : MODE 7
00020 RH=100 : D=1000 : CX=320 : CY=550
00024 FOR IR=75 TO 150 STEP 25 : RH=IR
00025   FOR IZ=4 TO 11 STEP 2 : TH=IZ*.1
00026     FOR IP=5TO15 STEP 2 : RESTORE : PH=IP*.1 : CLS
00027       PRINT@(0,0)"VIEW TH=";TH,"PH=";PH,"RH=";RH;
00030       S1=SIN(TH) : C1=COS(TH)
00040       S2=SIN(PH) : C2=COS(PH)
00050       GOTO 350
00060       REM PERSPECTIVE PROGECTION
00070       GOSUB 10000 : REM READ X,Y,Z
00080       XE=-X*S1+Y*C1
00090       YE=-X*C1*C2-Y*S1*C2+Z*S2
00100       ZE=-X*S2*C1-Y*S2*S1-Z*C2+RH
00110       SX%=D*XE/ZE+CX
00120       SY%=CY-D*YE/ZE
00125       RETURN
00130       REM HIDDEN LINE ROUTINE
00140       GOSUB 70 : XP=SX% : YP=SY%
00150       GOSUB 70 : XQ=SX% : YQ=SY%
00160       GOSUB 70 : XV=SX% : YV=SY%
00170       GOSUB 70 : XH=SX% : YH=SY%
00180       V1=(XH-XV)*.5 : V2=(YH-YV)*.5
00190       U1=XP-XQ      : U2=YP-YQ
00200       XT=XV+V1      : YT=YV+V2
00210       FOR I=2 TO 7
00220         V1=SGN(0.+(U2*(XV-XQ)-U1*(YV-YQ))*(U2*(XT-XQ)-U1*(YT-YQ)))
00230         V1=V1*.5 : V2=V2*.5
00240         XT=XT+V1*V1 : YT=YT+V1*V2
```

```
00250      NEXT
00260      PLOT XV,YV TO XT,YT
00270      RETURN
00280      REM VISABLE LINE ROUTINE
00290      FOR I=1 TO N%
00300        GOSUB 70
00310        IF I=1 THEN PLOT SX%,SY%
00320        PLOT TO SX%,SY%
00330      NEXT I
00335      RETURN
00340      REM DATA FOR HOUSE
00350      N%=11 : GOSUB 290
00360      DATA -11,-20, 23,  6,-20,  9,  6,-20,  0
00370      DATA   6, 32,  0,  0, 32,  0,  0, 32, 20
00380      DATA   0,  0, 20,  0,  0, 13,  0, 32, 13
00390      DATA   6, 32,  9,  6,-20,  9
00400      N%=2 : GOSUB 290
00410      DATA   6, 32,  0,  6, 32,  9
00420      N%=2 : GOSUB 290
00430      DATA   6,  0,  0,  6,  0,  9
00440      N%=3 : GOSUB 290
00450      DATA   0, 32,  0,  0,  0,  0,  6,  0,  0
00460      N%=5 : GOSUB 290
00470      DATA   0, 32,  0,-11.5,32, 0,-11.5,33.5,0
00480      DATA-11.5,33.5,33 ,-11.5,32,33
00490      GOSUB 140 : PLOT XP,YP TO XQ,YQ
00500      DATA -11.5,32,33, -11.5,32,0
00510      DATA   0,32,20,  -14,32,31.5
00520      GOSUB 160
00530      DATA -14,0,31.5  ,-14,32,31.5
00540      GOSUB 140 : PLOT XP,YP TO XQ,YQ
00550      DATA 0,0,20, -14,0,31.5
00560      DATA -11,-20,23, -11,0,23
00570      N%=2 : GOSUB 290
00580      DATA -11.5,33.5,0, -16.5,33.5,0
00590      N%=2 : GOSUB 290
00600      DATA -11.5,33.5,33, -16.5,33.5,33
00610      GOSUB 140 : PLOT XP,YP TO XQ,YQ
00620      DATA -16.5,33.5,0, -16.5,33.5,33
00630      DATA -28,32,0, -16.5,32,0
00640      GOSUB 160
00650      DATA -28,32,20, -14,32,33
00660      N%=2 : GOSUB 290
00670      DATA -28,32,20, -28,32,0
```

```
00680        GOSUB 140
00690        DATA  6,-20,9,  6,32,9
00700        DATA  0,0,0,    0,0,10
00710      FOR I=1 TO 5 : GOSUB 10000 : GOSUB 800 : NEXT I
00720        DATA 0,5,13.5, 0,14,13.5, 0,23,13.5
00730        DATA 0,14,2,  0,23,2
00740      FOR I=1 TO 2 : GOSUB 10000 : GOSUB 920 : NEXT I
00750        DATA 6,-11,0,  6,-1.5,0
00760        N%=4 : GOSUB 290
00770        DATA 0,6,0,  0,6,6.5
00780        DATA 0,9,6.5,  0,9,0
00787        TRONX
00788      NEXT IP
00789   NEXT IZ
00790 NEXT IR
00791 END
00800 REM WINDOWS SUBROUTINE
00810 GOSUB 80 : PLOT SX%,SY%
00820 Y=Y+4 : GOSUB 80 : PLOT TO SX%,SY%
00830 Z=Z+5.5:GOSUB 80 : PLOT TO SX%,SY%
00840 Y=Y-4 : GOSUB 80 : PLOT TO SX%,SY%
00850 Z=Z-5.5:GOSUB 80 : PLOT TO SX%,SY%
00860 Y=Y+2 : GOSUB 80 : PLOT SX%,SY%
00870 Z=Z+5.5:GOSUB 80 : PLOT TO SX%,SY%
00880 Y=Y-2 : Z=Z-2.75 : GOSUB 80 : PLOT SX%,SY%
00890 Y=Y+4 : GOSUB 80 : PLOT TO SX%,SY%
00900 RETURN
00910 REM GARAGE DOOR ROUTINE
00920 GOSUB 80 : PLOT SX%.SY%
00930 Z=Z+7 : GOSUB 80 : PLOT TO SX%,SY%
00940 Y=Y-7.5:GOSUB 80 : PLOT TO SX%,SY%
00950 Z=Z-7 : GOSUB 80 : PLOT TO SX%,SY%
00960 Z=Z+5 : Y=Y+1 : GOSUB 80 : PLOT SX%,SY%
00970 Z=Z+1 : GOSUB 80 : PLOT TO SX%,SY%
00980 Y=Y+5.5:GOSUB 80 : PLOT TO SX%,SY%
00990 Z=Z-1 : GOSUB 80 : PLOT TO SX%,SY%
01000 Y=Y-5.5:GOSUB 80 : PLOT TO SX%,SY%
01010 RETURN
10000 READ X,Y,Z : RETURN
10001 END
```

리스트 9. 12 3차원 그래픽으로 집을 그리는 프로그램

```
00010 REM***********************************
00020 REM** PYRAMID    3D GRAPHIC PYRAMID **
00030 REM** CREATED 07/05/84 BY A.G.      **
00040 REM***********************************
00050 DEFDBL A-Z : DEFINT A,D,I,J,K : CLS : PRINT"CALC.";
00060 DIM SP%(912),E%(6,3),V(4,3),SV%(4,2),S%(4,4),N(4,3)
00070 RH=15 : D=4000 : AD=1 : CX%=400 : CY%=500
00080 S1=SIN(.5) : C1=COS(.5) : S2=SIN(.9) : C2=COS(.9)
00090 CT=COS(.1) : ST=SIN(.1) : SO=SIN(-.1): CO=COS(-.1)
00100 SP=SIN(-.1): CP=COS(-.1)
00110 DATA   0, 0, 1.75
00120 DATA   1, 0, 0
00130 DATA -.2, 1, 0
00140 DATA -.2,-1, 0
00150 FOR I=1 TO 4
00160   READ X,Y,Z : V(I,1)=X : V(I,2)=Y : V(I,3)=Z : GOSUB 890
00170 NEXT
00180 DATA 1,4,2,1
00190 DATA 1,2,3,1
00200 DATA 1,3,4,1
00210 DATA 2,4,3,2
00220 FOR I=1 TO 4
00230   FOR J=1 TO 4
00240     READ S%(I,J)
00250   NEXT J
00260 NEXT I
00270 FOR IR = 1 TO 36
00280   FOR I=1 TO 6
00290     E%(I,3)=0
00300   NEXT I
00310   FOR I=1 TO 4
00320     U1=V(S%(I,2),1)-V(S%(I,1),1)
00330     U2=V(S%(I,2),2)-V(S%(I,1),2)
00340     U3=V(S%(I,2),3)-V(S%(I,1),3)
00350     V1=V(S%(I,3),1)-V(S%(I,1),1)
00360     V2=V(S%(I,3),2)-V(S%(I,1),2)
00370     V3=V(S%(I,3),3)-V(S%(I,1),3)
00380     N(I,1)=U2*V3-V2*U3:N(I,2)=U3*V1-V3*U1:N(I,3)=U1*V2-V1*U2
00390   NEXT I
00400   XE=RH*S2*C1 : YE=RH*S2*S1 : ZE=RH*C2
00410   N%=1
00420   FOR I=1 TO 4
```

```
00430     E2%=S%(I,1)
00440     WX=XE-V(E2%,1) : WY=YE-V(E2%,2) : WZ=ZE-V(E2%,3)
00450     LONGIF N(I,1)*WX+N(I,2)*WY+N(I,3)*WZ > 0
00460       E1%=S%(I,1)
00470       FOR J=2 TO 4
00480         E2%=S%(I,J)
00490         FOR K=1 TO N%
00500           IF E%(K,1)=E2% AND E%(K,2)=E1% THEN E%(K,3)=2 : GOTO 540
00510         NEXT K
00520         E%(N%,1)=E1% : E%(N%,2)=E2% : E%(N%,3)=1
00530         N%=N%+1
00540         E1%=E2%
00550       NEXT J
00560     ENDIF
00570   NEXT I
00580   FOR I=1 TO 6
00590     LONGIF E%(I,3)
00600       J=E%(I,1) : K=E%(I,2)
00610       SP%(AD  )=SV%(J,1) : SP%(AD+1)=SV%(J,2)
00620       SP%(AD+2)=SV%(K,1) : SP%(AD+3)=SV%(K,2)
00630     ENDIF
00640     AD=AD+4
00650   NEXT
00660   FOR I=1 TO 4
00670     T1=CP*CT*V(I,1)-(ST*CP+SO*SP)*V(I,2)+(SO*ST*CP-SP*CO)*V(I,3)
00680     T2=ST*V(I,1)+CO*CT*V(I,2)-SO*CT*V(I,3)
00690     T3=SP*CT*V(I,1)+(SO*CP-CO*ST*SP)*V(I,2)+(ST*SO*SP+CO*CP)*V(I,3)
00700     V(I,1)=T1 : V(I,2)=T2 : V(I,3)=T3 : X=T1 : Y=T2 : Z=T3
00710     GOSUB 890
00720   NEXT
00730   TRONB:TROFF:PRINT"*";
00740 NEXT
00750 FOR I=1 TO 48
00760   SP%(I+864)=SP%(I)
00770 NEXT I
00780 AD=1 : M=(M+1)AND7 : MODE M : CLS
00784 PRINT@(0,0);"MODE=";M
00785 PRINT@(0,1);"TIME=";TIME$;
00786 PRINT@(0,2);"DATE=";DATE$;
00790 COLOR -1  : GOSUB 820 : DELAY 25 : COLOR 0 : AD=AD-24 : GOSUB 820
00800 TRONB:TROFF:IF AD=865 THEN 780
00810 GOTO 784
00820 FORI=1 TO 6
00830   LONGIF SP%(AD)
```

```
00840      PLOT SP%(AD),SP%(AD+1) TO SP%(AD+2),SP%(AD+3)
00850   ENDIF
00860   AD=AD+4
00870 NEXT
00880 RETURN
00890 XE=-X*S1+Y*C1
00900 YE=-X*C1*C2-Y*S1*C2+Z*S2
00910 ZE=-X*S2*C1-Y*S2*S1-Z*C2+RH
00920 SV%(I,1)= D*XE/ZE+CX%
00930 SV%(I,2)=-D*YE/ZE+CY%
00940 RETURN
00941 END
```

리스트 9. 13 3차원으로 피라미드를 각 모드별로 그린다.

부록

애플 IIe의 응용

§1. WINDOW MASTER
§2. 128K VOICE
§3. 스크린 스위치

§1. WINDOW MASTER

1. 개요

프로그램을 실행시키는 도중이거나 작성하는 도중에, 화면에 어떤 메시지를 출력하거나 작업을 하려면 어떻게 할 것인가? 아마 화면의 전부 또는 일부를 지운 다음 원하는 일을 할 것이다. 그러나 원래의 화면을 지우지 않은 채로 작업을 하고자 할 때 이 프로그램은 놀라운 위력을 발휘한다.

원래의 화면을 그대로 보존한 채 사용자가 지정한 윈도우를 만든다. 그 다음 작업을 끝낸 후 윈도우를 지우면 화면이 본래대로 출력된다. 그리고 이 프로그램의 큰 장점은 반전된 윈도우도 만들 수 있다는 것이다. 이 프로그램은 80컬럼을 내장한 애플Ⅱe에서만 실행 가능하며 DOS3.3과 ProDOS상에서 모두 사용할 수 있다.

2. 입력 방법과 세이브

머린 어셈블러가 있다면 리스트1을 입력하고, 없다면 리스트2의 기계어 덤프 리스트를 입력한다. 리스트 3은 데모 프로그램이므로 애플소프트 베이직상에서 입력하면 된다. 디스크에 세이브하려면 메인 기계어 프로그램은

```
BSAVE WINDOW MASTER, A$94BE, L$142
```

베이직 데모 프로그램은

```
SAVE WINDOW MASTER DEMO
```

로 세이브하면 된다.

3. 로드와 사용 방법

리스트2는 DOS상에서

```
BLOAD WINDOW MASTER
```

하면 되고 리스트3은

LOAD WINDOW MASTER DEMO

하면 된다.

그리고 사용하기 전에 DOS3.3상이면 HIMEM : 36864를 지정해 주고, ProDOS상이면 HIMEM : 37888 을 지정해 준다.

그러면 반전되지 않는 윈도우(normal window)와 반전되는 윈도우(inverse window)로 나누어 설명하겠다.

(1) NORMAL WINDOW

베이직상에서 CALL WINDOW, VS, VE, HS, HE, 0의 형식을 사용한다. 이전에 PR#3으로 반드시 80컬럼 모드로 되어 있어야만 하며, WINDOW, VS, VE, HS, HE의 값도 정해져 있어야만 한다.

WINDOW는 CALL 번지로 WINDOW = 388116이며 VS, VE, HS, HE의 값에 대한 것을 그림 10.1에 나타내었다.

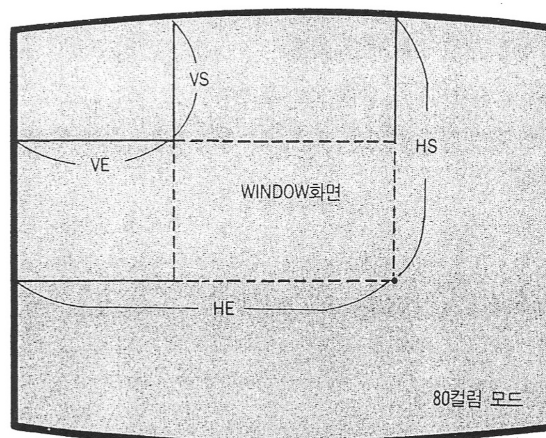


그림 10.1 window 화면

VS, VE, HS, HE의 범위는 다음과 같다. VS는 1~23, VE는 2~24, HS는 1~78, HE는 3~80 사이의 값은 사용하며 제한된 숫자 이외의 것은 사용하면 SYNTAX ERROR가 발생한다.

윈도우를 지우려면 CALL WINDOW, VS, VE, HS, HE, 1을 사용한다. 여기에서 WINDOW, VS, VE, HS, HE의 값은 이전에 지정된 값과 동일해야만 한다. 메시지의 출력이나 작업은 윈도우 안에서만 해야 한다.

(2) INVERSE WINDOW

베이직상에서 CALL INVRS, VS, VE, HS, HE, 0의 형식을 사용한다. INVRS는 CALL번지로 INVRS=38078이며 VS, VE, HS, HE의 값에 대한 것과 윈도우를 지우는 것은 NORMAL WINDOW와 완전히 같다.

4. 프로그램의 설명

(1) 어셈블리어 프로그램에 대한 설명

1~ 10 : 코멘트
11~ 41 : EQUATE(어드레스와 값들을 지정)
42~ 43 : ORG값을 지정($94BE)
44~ 48 : SPACE를 반전
49~ 57 : 윈도우를 반전하고 밑줄을 만듬.
58 : 베이직으로 돌아감.
59~ 69 : 다섯 개의 파라메터값을 체크함.
70~ 72 : 80컬럼 모드인가를 체크하고 아니면 에러 발생.
73~ 87 : 제한 범위의 값을 벗어나면 에러 발생.
88~ 94 : 윈도우를 만들 것인가, 지울 것인가를 체크함.
95~176 : 윈도우를 만들 때 원래의 화면을 보관하고 윈도우를 지울 때 원래의 화면을 출력함.
177 : TMPVMODE
178 : VSBOX
179 : CPFLAG
180 : HZEND, HZSTRT, VTEND, VTSTRT

(2) 베이직 프로그램에 대한 설명

10~ 70 : 코멘트
80 : 80컬럼 모드 세트
90 : 메시지 출력
100 : 에러 발생시 240행으로 점프
110 : 메인 기계어 프로그램을 로드
120 : WINDOW, INVRS의 값을 정의
130~180 : NORMAL WINDOW 데모

190~230 : INVERSE WINDOW 데모
240 : 에러시 메시지 출력

리스트 10.1

```
:ASM
        ************************
        *                      *
        *    WINDOW MASTER     *
        *                      *
        *       BY RYU.HS      *
        *                      *
        * COMPUTER STUDY       *
        *              PC CLUB *
        *                      *
        ************************
        *
        *  EQUATES
        *
        NUMCHRS  =    $00
        BUFFLO   =    $01
        BUFFHI   =    $02
        TEMPHI   =    $03
        LINNUM   =    $50
        STREND   =    $6E
        FRETOP   =    $70
        VMODE    =    $4FB
        OURCH    =    $57B
        CLRCHR   =    $956A
        BOXTOP   =    $95AB
        BOXSIDE  =    $95C6
        BOXBTM   =    $95E5
        TMPVMODE =    $95F9
        VSBOX    =    $95FA
        CPFLAG   =    $95FB
        HZEND    =    $95FC
        HZSTRT   =    $95FD
        VTEND    =    $95FE
        VTSTRT   =    $95FF
        RD80STOR =    $C018
        STORE    =    $CE38
        PICK     =    $CE44
        FRMNUM   =    $DD67
```

```
                CHKCOM   =    $DEBE
                SNERR    =    $DEC9
                GETADR   =    $E752
                BASCALC  =    $FBC1
                *
                         ORG  $94BE
94BE: A9 20     INVRS    LDA  #$20
94C0: 8D 6A 95           STA  CLRCHR
94C3: 8D AB 95           STA  BOXTOP
94C6: 8D C6 95           STA  BOXSIDE
94C9: 8D E5 95           STA  BOXBTM
94CC: 20 E4 94           JSR  WINDOW
94CF: A9 A0              LDA  #$A0
94D1: 8D 6A 95           STA  CLRCHR
94D4: A9 4C              LDA  #$4C
94D6: 8D AB 95           STA  BOXTOP
94D9: A9 FC              LDA  #$FC
94DB: 8D C6 95           STA  BOXSIDE
94DE: A9 DF              LDA  #$DF
94E0: 8D E5 95           STA  BOXBTM
94E3: 60                 RTS
                *
94E4: A2 05     WINDOW   LDX  #05
94E6: 86 00     W1       STX  NUMCHRS
94E8: 20 BE DE           JSR  CHKCOM
94EB: 20 67 DD           JSR  FRMNUM
94EE: 20 52 E7           JSR  GETADR
94F1: A5 50              LDA  LINNUM
94F3: A6 00              LDX  NUMCHRS
94F5: 9D FA 95           STA  VSBOX,X
94F8: CA                 DEX
94F9: D0 EB              BNE  W1
94FB: AD 18 C0           LDA  RD80STOR
94FE: 29 80              AND  #$80
9500: F0 21              BEQ  ERROR
9502: AD FF 95           LDA  VTSTRT
9505: CD FE 95           CMP  VTEND
9508: B0 19              BCS  ERROR
950A: AD FE 95           LDA  VTEND
950D: C9 19              CMP  #$19
950F: B0 12              BCS  ERROR
9511: 18                 CLC
9512: AD FD 95           LDA  HZSTRT
9515: 69 01              ADC  #$01
9517: CD FC 95           CMP  HZEND
```

```
951A: B0 07          BCS  ERROR
951C: AD FC 95       LDA  HZEND
951F: C9 51          CMP  #$51
9521: 90 03          BCC  W2
9523: 4C C9 DE  ERROR  JMP  SNERR
9526: A9 00     W2     LDA  #0
9528: 85 01          STA  BUFFLO
952A: AD FB 95       LDA  CPFLAG
952D: F0 06          BEQ  W3
952F: A5 03          LDA  TEMPHI
9531: 85 02          STA  BUFFHI
9533: 10 0A          BPL  W4
9535: 18        W3     CLC
9536: A5 6E          LDA  STREND
9538: 65 70          ADC  FRETOP
953A: 4A             LSR
953B: 85 02          STA  BUFFHI
953D: 85 03          STA  TEMPHI
953F: CE FD 95  W4     DEC  HZSTRT
9542: CE FF 95       DEC  VTSTRT
9545: AD FF 95       LDA  VTSTRT
9548: 8D FA 95       STA  VSBOX
954B: AD FF 95       LDA  VTSTRT
954E: 20 C1 FB  W5     JSR  BASCALC
9551: AC FD 95       LDY  HZSTRT
9554: 8C 7B 05  W6     STY  OURCH
9557: AD FB 95       LDA  CPFLAG
955A: F0 06          BEQ  CLEAR
955C: A4 00          LDY  NUMCHRS
955E: B1 01          LDA  (BUFFLO),Y
9560: D0 09          BNE  W7
9562: 20 44 CE  CLEAR  JSR  PICK
9565: A4 00          LDY  NUMCHRS
9567: 91 01          STA  (BUFFLO),Y
9569: A9 A0          LDA  #$A0
956B: AC 7B 05  W7     LDY  OURCH
956E: 20 38 CE       JSR  STORE
9571: E6 00          INC  NUMCHRS
9573: C8             INY
9574: CC FC 95       CPY  HZEND
9577: D0 DB          BNE  W6
9579: 18             CLC
957A: A5 01          LDA  BUFFLO
957C: 65 00          ADC  NUMCHRS
957E: 85 01          STA  BUFFLO
```

```
9580: 90 02      128            BCC  W8
9582: E6 02      129            INC  BUFFHI
9584: A9 00      130  W8        LDA  #0
9586: 85 00      131            STA  NUMCHRS
9588: EE FF 95   132            INC  VTSTRT
958B: AD FF 95   133            LDA  VTSTRT
958E: CD FE 95   134            CMP  VTEND
9591: D0 BB      135            BNE  W5
9593: AD FB 95   136            LDA  CPFLAG
9596: F0 01      137            BEQ  BOX
9598: 60         138            RTS
                 139  *
9599: AD FB 04   140  BOX       LDA  VMODE
959C: 8D F9 95   141            STA  TMPVMODE
959F: 29 FE      142            AND  #$FE
95A1: 8D FB 04   143            STA  VMODE
95A4: AD FA 95   144            LDA  VSBOX
95A7: 20 C1 FB   145            JSR  BASCALC
95AA: A9 4C      146            LDA  #$4C
95AC: 20 E6 95   147            JSR  DRAW
95AF: AD FB 04   148            LDA  VMODE
95B2: 09 01      149            ORA  #$01
95B4: 8D FB 04   150            STA  VMODE
95B7: CE FC 95   151            DEC  HZEND
95BA: AD FA 95   152  SIDES     LDA  VSBOX
95BD: CD FE 95   153            CMP  VTEND
95C0: F0 16      154            BEQ  BOTTOM
95C2: 20 C1 FB   155            JSR  BASCALC
95C5: A9 FC      156            LDA  #$FC
95C7: AC FD 95   157            LDY  HZSTRT
95CA: 20 38 CE   158            JSR  STORE
95CD: AC FC 95   159            LDY  HZEND
95D0: 20 38 CE   160            JSR  STORE
95D3: EE FA 95   161            INC  VSBOX
95D6: D0 E2      162            BNE  SIDES
95D8: CE FE 95   163  BOTTOM    DEC  VTEND
95DB: EE FD 95   164            INC  HZSTRT
95DE: AD FE 95   165            LDA  VTEND
95E1: 20 C1 FB   166            JSR  BASCALC
95E4: A9 DF      167            LDA  #$DF
95E6: AC FD 95   168  DRAW      LDY  HZSTRT
95E9: 20 38 CE   169  D1        JSR  STORE
95EC: C8         170            INY
95ED: CC FC 95   171            CPY  HZEND
95F0: D0 F7      172            BNE  D1
```

```
95F2: AD F9 95          LDA  TMPVMODE
95F5: 8D FB 04          STA  VMODE
95F8: 60                RTS
                   *
95F9: 00                HEX  00
95FA: 00                HEX  00
95FB: 00                HEX  00
95FC: 00 00 00          HEX  00000000
95FF: 00


--End assembly--

322 bytes

Errors: 0


Symbol table - alphabetical order:

   BASCALC =$FBC1     BOTTOM  =$95D8     BOX     =$9599     BOXBTM  =$95E5
   BOXSIDE =$95C6     BOXTOP  =$95AB     BUFFHI  =$02       BUFFLO  =$01
   CHKCOM  =$DEBE     CLEAR   =$9562     CLRCHR  =$956A     CPFLAG  =$95FB
   D1      =$95E9     DRAW    =$95E6     ERROR   =$9523     FRETOP  =$70
   FRMNUM  =$DD67     GETADR  =$E752     HZEND   =$95FC     HZSTRT  =$95FD
?  INVRS   =$94BE     LINNUM  =$50       NUMCHRS =$00       OURCH   =$057B
   PICK    =$CE44     RD80STOR=$C018     SIDES   =$95BA     SNERR   =$DEC9
   STORE   =$CE38     STREND  =$6E       TEMPHI  =$03       TMPVMODE=$95F9
   VMODE   =$04FB     VSBOX   =$95FA     VTEND   =$95FE     VTSTRT  =$95FF
   W1      =$94E6     W2      =$9526     W3      =$9535     W4      =$953F
   W5      =$954E     W6      =$9554     W7      =$956B     W8      =$9584
   WINDOW  =$94E4


Symbol table - numerical order:

   NUMCHRS =$00       BUFFLO  =$01       BUFFHI  =$02       TEMPHI  =$03
   LINNUM  =$50       STREND  =$6E       FRETOP  =$70       VMODE   =$04FB
   OURCH   =$057B  ?  INVRS   =$94BE     WINDOW  =$94E4     W1      =$94E6
   ERROR   =$9523     W2      =$9526     W3      =$9535     W4      =$953F
   W5      =$954E     W6      =$9554     CLEAR   =$9562     CLRCHR  =$956A
   W7      =$956B     W8      =$9584     BOX     =$9599     BOXTOP  =$95AB
   SIDES   =$95BA     BOXSIDE =$95C6     BOTTOM  =$95D8     BOXBTM  =$95E5
   DRAW    =$95E6     D1      =$95E9     TMPVMODE=$95F9     VSBOX   =$95FA
   CPFLAG  =$95FB     HZEND   =$95FC     HZSTRT  =$95FD     VTEND   =$95FE
```

```
VTSTRT  =$95FF      RDBOSTOR=$C018      STORE   =$CE38      PICK    =$CE44
FRMNUM  =$DD67      CHKCOM  =$DEBE      SNERR   =$DEC9      GETADR  =$E752
BASCALC =$FBC1
```

리스트 10.2

```
94BE- A9 20
94C0- 8D 6A 95 8D AB 95 8D C6
94C8- 95 8D E5 95 20 E4 94 A9
94D0- A0 8D 6A 95 A9 4C 8D AB
94D8- 95 A9 FC 8D C6 95 A9 DF
94E0- 8D E5 95 60 A2 05 86 00
94E8- 20 BE DE 20 67 DD 20 52
94F0- E7 A5 50 A6 00 9D FA 95
94F8- CA D0 EB AD 18 C0 29 80
9500- F0 21 AD FF 95 CD FE 95
9508- B0 19 AD FE 95 C9 19 B0
9510- 12 18 AD FD 95 69 01 CD
9518- FC 95 B0 07 AD FC 95 C9
9520- 51 90 03 4C C9 DE A9 00
9528- 85 01 AD FB 95 F0 06 A5
9530- 03 85 02 10 0A 18 A5 6E
9538- 65 70 4A 85 02 85 03 CE
9540- FD 95 CE FF 95 AD FF 95
9548- 8D FA 95 AD FF 95 20 C1
9550- F8 AC FD 95 8C 7B 05 AD
9558- FB 95 F0 06 A4 00 B1 01
9560- D0 09 20 44 CE A4 00 91
9568- 01 A9 A0 AC 7B 05 20 38
9570- CE E6 00 C8 CC FC 95 D0
9578- DB 18 A5 01 65 00 85 01
9580- 90 02 E6 02 A9 00 85 00
9588- EE FF 95 AD FF 95 CD FE
9590- 95 D0 BB AD FB 95 F0 01
9598- 60 AD FB 04 8D F9 95 29
95A0- FE 8D FB 04 AD FA 95 20
95A8- C1 FB A9 4C 20 E6 95 AD
95B0- FB 04 09 01 8D FB 04 CE
95B8- FC 95 AD FA 95 CD FE 95
95C0- F0 16 20 C1 FB A9 FC AC
95C8- FD 95 20 38 CE AC FC 95
95D0- 20 38 CE EE FA 95 D0 E2
95D8- CE FE 95 EE FD 95 AD FE
95E0- 95 20 C1 FB A9 DF AC FD
95E8- 95 20 38 CE C8 CC FC 95
95F0- D0 F7 AD F9 95 8D FB 04
95F8- 60 00 00 00 00 00 00 00
```

리스트 10.3

```
10  REM  ******************************
20  REM  *                            *
30  REM  *     WINDOW MASTER DEMO     *
40  REM  *                            *
50  REM  *          BY RYU.HS         *
60  REM  *                            *
70  REM  ******************************
80  PRINT  CHR$ (4)"PR#3"
90  PRINT "WINDOW MASTER DEMONSTRATION": PRINT
   "BY RYU.HS": PRINT "COMPUTER STUDY PC C
   LUB": PRINT
100  PRINT : ONERR  GOTO 240
110  PRINT  CHR$ (4)"BLOAD WINDOW MASTER":
```

```
    HIMEM: 36864
120  POKE 216,0:WINDOW = 38116:INVRS = 380
    78: LIST
130  REM    NORMAL WINDOW DEMO
140 VS = 3:VE = 10:HS = 10:HE = 60: CALL W
    INDOW,VS,VE,HS,HE,0
150  VTAB 4: POKE 1403,15: PRINT "This win
    dow was invokked from BASIC by:": VTAB
    6: POKE 1403,20: PRINT "CALL WINDOW,VS,
    VE,HS,HE,0"
160  VTAB 9: POKE 1403,13: PRINT "Press an
    y key for INVERSE WINDOW or ";: INVERSE
    : PRINT "Q";: NORMAL : PRINT "uit: ";: GET
    A$
170  CALL WINDOW,VS,VE,HS,HE,1
180  IF A$ =  CHR$ (81) OR A$ =  CHR$ (113
    ) THEN  VTAB 23: END
190  REM    INVERSE WINDOW DEMO
200 VS = 12:VE = 18:HS = 1:HE = 75: CALL I
    NVRS,VS,VE,HS,HE,0
210  VTAB 14: POKE 1403,15: INVERSE : PRINT
    "This INVERSE window was invoked from B
    ASIC by:": VTAB 16: POKE 1403,24: PRINT
    "CALL INVRS,VS,VE,HS,HE,0": NORMAL
220  VTAB 24: POKE 1403,0: GET A$: CALL WI
    NDOW,VS,VE,HS,HE,1
230  GOTO 130
240  HOME : PRINT "WINDOW MASTER OBJECT FI
    LE NOT FOUND.": END
```

§ 2. 128K VOICE

Ⅱe의 방대한 메모리 128K. 이 많은 메모리를 더블 고해상도 그래픽을 사용하지 않을 때는 어떻게 사용할 것인가? 그래서 남는 메모리를 유용하게 쓸 수 있는 음성 합성(소프트웨어만으로)을 구상해 보았다. 여기에 쓰이는 음성 합성의 방법은 지극히 간단한 것으로, 음성 데이타를 카세트로부터 수집하여 애플Ⅱe의 내장 스피커로 재생시키는 것이다. 이 방법은 Robert A. Payne의 저서 "A Voice For The Apple Ⅱ Without Extra Hardware"에서 인용한 것임을 먼저 밝혀둔다. 이 방법은 순전히 소프트웨어만으로 작동되므로 경비가 들지 않는 반면 음질은 떨어진다.

1. 음성 데이타의 수집

음성이 녹음되어 있는 테이프를 카세트에 넣고 카세트의 EAR단자와 Ⅱe의 카세트 IN 단자를 연결시키고 카세트를 플레이시키면 테이프에 담긴 육성이 어느 기준점을 지나면서 변화하는 것을 Ⅱe로 알 수가 있다. 이것은 Ⅱe의 $C060번지의 비트7을 검사해 보면 알 수 있다. 기준점을 기준으로 육성을 1 또는 0의 데이타로 받아들이게 된다. 여기서 그 기준점을 Threshold라 한다. 이러한 원리로 Ⅱe는 프로그램을 카세트 테이프에 LOAD/SAVE를 할 수 있는 것이다. 그림10.2를 보면 쉽게 이해될 것이다.

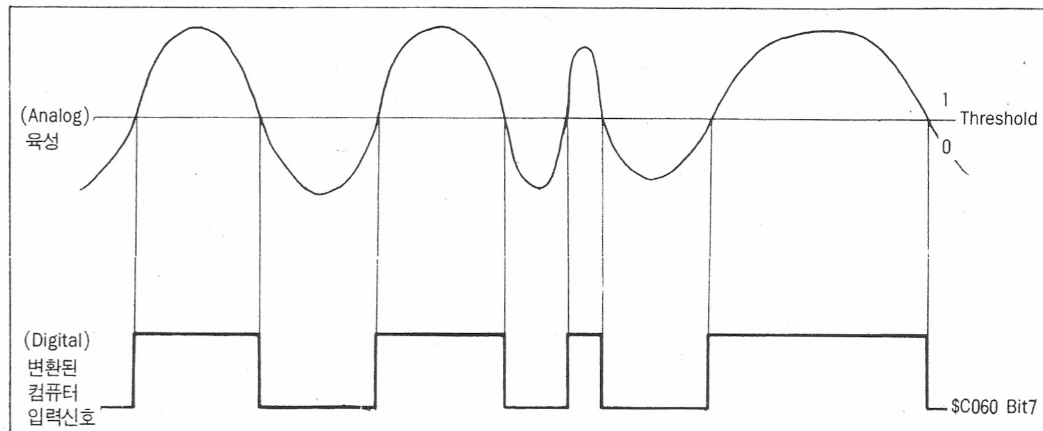


그림10.2 음성 신호

이러한 방법으로 음성 수집을 하게 되는데, 육성의 특성만 저장이 되므로 음질이 당연히 떨어지게 되지만 어쩔 수 없다.

음성 데이타 수집에 쓰인 VOICE 프로그램에서는 각 8비트 메모리중 첫째 비트(b7)만을 데이타 비트로 사용하므로 한 개의 비트만 A/D변환된 상태로 저장하고 나머지 일곱 비트는 변화하지 않고 유지되는 시간만큼을 카운트하기 위한 카운터로 사용하였다.

저주파인 경우 실제로 샘플링(sampling)할 때마다 데이타가 변화하지는 않으므로 8비트 전체를 데이타로 쓰는 것보다 메모리를 훨씬 절약할 수 있다.

만약에 카운터가 가득 차서 Overflow되었다면 다음 메모리 장소에 똑같은 데이타 값("0"이나 "1")이 새로운 카운터 값과 함께 저장될 것이다. 카운터는 7비트이므로 $0~$7F의 정보를 담을 수 있을 것이다.

2. 음성의 재생

애플Ⅱe의 스피커는 $C030번지를 Read 또는 Write함으로써 한 순간 신호를 받게 된다. 메모리에

저장된 음성 데이타인 1비트 데이타+ 7비트 카운터를 사용하여 그 카운터 값이 0으로 되고 저장된 데이타 값이 변화(0↔1)할 때마다 스피커에 신호를 주면, 녹음되었던 소리와 동일한 특성을 가진 소리를 들을 수 있을 것이다.

간단히 말하면 원래 육성에 대응되는 주파수가 그대로 출력되는 연속인 신호들을 예로 든다면 어떤 긴밀한 관계가 있다는 것을 알게 될 것이다. 이 변화하는 주파수가 바로 입력 신호(육성)에 대한 FM 변환인 것이다.

3. 프로그램 구성

128K VOICE는 ProDOS상에서 램디스크를 활용하는 프로그램으로 LOADER, PLAYER로 크게 구성되고 메인 부분은 기계어로 VOICE라고 따로 나누어지며 타이틀 부분도 따로 나누어져 있다. 프로그램들의 입력, 세이브 방법 등은 뒤에서 다루기로 하고, 먼저 128K VOICE가 실행되면 멋진 타이틀(타이틀만 따로 뽑아 다른 프로그램에 쓸 수도 있음 : 그림10.3 참조) 화면과 함께 음악이 나온다. 아무 키나 누르게 되면 메뉴 화면이 나오는데 LOADER, PLAYER 둘 중에 하나를 택해야 한다.

LOADER는 음성 데이타를 메모리로 수집하는 것으로 실행시킨 후 음성 수집한 테이프를 카세트에 넣고 카세트의 EAR 단자와 Ⅱe의 카세트 IN 단자와 연결하고 볼륨은 ⅔정도에 두고 PLAY를 누르고 스페이스바를 누른다. 그리고 수집한 데이타를 로드하게 된다. 이 LOADER로 약 1분 정도의 음성 데이타를 수집할 수가 있는데 수집할 때 샘플링 딜레이인 VOICE 루틴의 $81F 부분을 수정하면

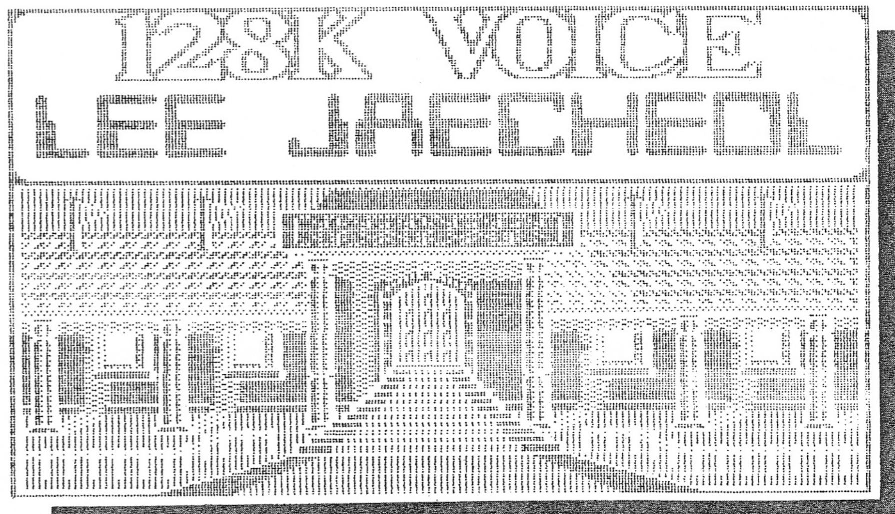


그림 10.3 128K VOICE의 타이틀 화면

샘플링 속도를 빠르게 또는 늦출 수도 있다. 이 샘플링 속도는 음성 수집된 데이타의 음질을 좌우하게 된다.

PLAYER는 음성 데이타를 다시 재생하는 것으로서 128K 메모리에서는 ProDOS의 램디스크를 이용, 앞에서 LOADER로 수집한 데이타 두 개를 읽어들인 후 하나를 램디스크에 저장시키고 또 하나는 메인 메모리에 저장시켜 약 2분간의 음성 데이타를 재생시키게 된다. 보통 64K 애플 수집량의 두 배이다(그래서 제목을 128K VOICE라 하였다).

기계어 루틴인 VOICE에서 $800~$859까지는 LOADER 부분이고 $860~$8A5까지가 PLAYER 부분인데 이 $877의 딜레이를 조정함으로써 실행 속도를 조정할 수 있게 되며 그 속도가 빠를수록 음질은 소프라노 쪽으로, 느릴수록 베이스 쪽으로 가까워질 것이다. 정확히 조정하면 좋은 음질을 얻을 수도 있으니 LOADER를 사용할 때 카세트의 볼륨, 샘플링 속도, 그리고 PLAYER의 실행 속도, 이 세 가지를 조화있게 잘 조정해 보기 바란다.

만약 유저가 샘(SAM)카드를 가지고 있는 경우는 VOICE 루틴의 PLAYER 부분인 $873을 8D 30 C0에서 8D A0 C0으로 고쳐서 실행시킨다면 더욱 훌륭한 음질을 들을 수 있게 된다.

4. 프로그램 입력 방법

128K VOICE는 ProDOS의 램디스크를 사용하므로 ProDOS가 부팅된 상태에서 입력해야 한다. 공디스켓에 먼저 ProDOS로 포매팅시켜 놓고 아래 리스트들을 다음과 같은 방법으로 입력한다.

- STARTUP– 베이직
- STARTUP2– 베이직
- LOADER– 베이직
- PLAYER– 베이직
- TIPIC– A$4000, L$5FFF
- MUSIC– A$1400, L$1FFF
- VOICE– A$800, L$A6

리스트 10.4

```
10  HOME
20  PRINT "128K VOICE(TM) PROGRAM DISK"
30  PRINT "Made by PC club chairman Lee jae cheol
."
40  VTAB 12: HTAB 12: PRINT "Wait a moment..."
```

```
 50  VTAB 13
 60  PRINT  CHR$ (4);"BLOAD MUSIC"
 70  POKE 230,32: CALL 62450: HGR : POKE  - 16302,
0
 80  PRINT  CHR$ (4);"BLOAD TIPIC,A8192"
 90  POKE 235,12: POKE 236,28
 100  CALL 5120
 105  POKE 103,1: POKE 104,10: POKE 2560,0
 110  PRINT  CHR$ (4);"RUN STARTUP2"
 65535  REM
```

리스트 10.5

```
 10  TEXT : HOME
 20  FLASH : PRINT "128K VOICE";: NORMAL : PRINT "
MENU"
 30  PRINT
 40  PRINT "1.DATA LOAD IN MEMORY"
 50  PRINT
 60  PRINT "2.DATA PLAY FROM MEMORY"
 70  PRINT
 80  INPUT "SELECT NUMBER:";N
 90  ON N GOTO 110,120,100
 100  END
 110  PRINT  CHR$ (4);"RUNLOADER"
 120  PRINT  CHR$ (4);"RUNPLAYER"
```

리스트 10.6

```
 10 D$ =  CHR$ (4): PRINT D$"BLOADVOICE"
 20  HOME : FLASH : PRINT "128K VOICE";: NORMAL :
PRINT " LOADER"
 30  PRINT : PRINT "PRESS PLAY AND SPACE BAR";: GE
T G$
 40  CALL 2048
 45  PRINT
 50  INPUT "SAVE FILE NAME:";F$
 60  PRINT D$"BSAVE"F$",A$1000,L$8000"
 70  INPUT "DATA LOAD AGAIN?(Y/N)";L$
 80  IF L$ = "Y" THEN 20
 90  PRINT "DONE.": END
```

리스트 10.7

```
 10 D$ =  CHR$ (4): PRINT D$"BLOADVOICE"
 20  HOME : FLASH : PRINT "128K VOICE";: NORMAL :
PRINT " PLAYER"
 30  PRINT "INSERT DATA DISK ";: GET G$
 35  PRINT
 40  INPUT "DATA2 FILE NAME:";F2$
 45  PRINT
 50  INPUT "DATA1 FILE NAME:";F1$
 60  PRINT D$"BLOAD"F2$
 70  PRINT D$"BSAVE/RAM/"F2$",A$1000,L$8000"
 80  PRINT D$"BLOAD"F1$
 90  CALL 2144
 95  PRINT D$"BLOAD/RAM/"F2$
 96  CALL 2144
 100  PRINT : INPUT "PLAY AGAIN?(Y/N)";O$
 110  IF O$ = "Y" THEN 20
 120  PRINT "DONE.": END
```

리스트 10.8

```
P:1  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4000- FF FF FF FF FF FF FF FF:F8
4008- FF FF FF FF FF FF FF FF:F8
4010- FF FF FF FF FF FF FF FF:F8
4018- FF FF FF FF FF FF FF FF:F8
4020- FF FF FF FF FF FF FF FF:F8
4028- 87 00 00 00 00 00 00 00:87
4030- 00 00 00 00 00 00 00 00:00
4038- 00 00 00 00 00 00 00 00:00
4040- 00 00 00 00 00 00 00 00:00
4048- 00 00 00 00 00 00 00 70:70
4050- F3 AC FC CF 88 80 E3 AC:01
4058- FC CF 88 80 E3 BF B7 FE:2A
4060- 99 BB BA DD AE EE FF BF:45
4068- B7 FE 9F 91 00 C4 FF AC:54
4070- FC CF 88 80 E3 AC FC E7:45
4078- 00 00 00 00 00 00 00 00:00
4080- 83 00 00 00 00 0C 78 1C:23
4088- 70 40 01 03 03 0F 58 73:91
4090- 00 00 00 60 40 43 33 30:46
4098- 00 03 07 1C 18 00 1C 0F:69
---------------------------------
SUM  B0 41 68 B7 52 F6 AE 35 :3B

P:2  +0 +1 +2 +3 +4 +5 +6 +7 SUM
40A0- 18 00 00 00 00 00 00 E0:F8
40A8- D3 AA D5 AA D5 AA D5 AA:FA
40B0- D5 AA D5 AA D5 AA D5 FF:51
40B8- FF FF FF FF FF FF FF FF:F8
40C0- D5 AA D5 AA D5 AA D5 AA:FC
40C8- D5 AA D5 AA D5 AA D5 E2:34
40D0- F3 EE FC CF 88 C0 E3 EE:C5
40D8- FC CF 88 C0 E3 BF B7 FE:6A
40E0- 99 BB BA DD AE EE FF BF:45
40E8- B7 FE 9F 91 80 C5 FF EE:17
40F0- FC CF 88 C0 E3 EE FC E7:C7
40F8- 00 00 00 00 00 00 00 00:00
4100- 83 00 00 00 00 0C 18 00:A7
4108- 07 0E 1E 06 70 0C 18 30:FD
4110- 00 00 00 00 0E 18 1F 30:75
4118- 00 03 06 0C 18 00 00 0C:39
4120- 78 63 00 00 00 00 00 E0:BB
4128- D3 AA D5 D5 CA AA D5 AA:1A
4130- D5 D5 D2 AA 85 80 80 80:2B
```

```
4138- 80 80 80 80 80 80 80 80:00
-----------------------------------
SUM  CE 5F 03 75 34 A1 0B 8A :0F

P:3  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4140- 80 80 D0 AA D5 D5 CA AA:98
4148- D5 AA D5 D5 D2 AA D5 E2:5C
4150- F3 EE FC CF FF FF CF EE:67
4158- FC CF FF FF CF BF 96 FE:EB
4160- 99 CB DF AA FD E9 FF BF:91
4168- 96 FE 9F FF FF 9F FF EE:BD
4170- FC CF FF FF CF EE FC E7:69
4178- 00 00 00 00 00 00 00 00:00
4180- 83 00 00 00 00 0C 58 03:EA
4188- 00 40 43 63 41 0D 18 0E:5A
4190- 18 00 00 00 70 70 60 61:B9
4198- 73 61 07 7C 70 71 63 0D:A8
41A0- 18 00 67 00 00 00 00 E0:5F
41A8- 33 3B 7B B9 3B 3B 3B 3B:8E
41B0- 7B B9 3B 3B E7 FB FF EF:7A
41B8- FF EF FF EF FF EF FF EF:B8
41C0- FF EF F3 EE 6E E9 6E 6E:02
41C8- 6E 6E 6E 69 6E 6E 6E 66:63
41D0- F3 EE FC CF F9 FF FF EE:91
41D8- FC CF F9 FF FF BF B7 FE:36
-----------------------------------
SUM  9E 1D D9 DC 56 E7 FC 44 :ED

P:4  +0 +1 +2 +3 +4 +5 +6 +7 SUM
41E0- 99 BF D5 AA D5 FE F9 BF:62
41E8- B7 FE 9F F3 FF FF FF EE:32
41F0- FC CF F9 FF FF EE FC E7:93
41F8- 00 00 00 00 00 00 00 00:00
4200- 83 00 00 00 00 00 00 00:83
4208- 00 00 00 00 00 00 00 00:00
4210- 00 00 00 00 00 00 00 00:00
4218- 00 00 00 00 00 00 00 00:00
4220- 00 00 00 00 00 00 00 E0:E0
4228- 33 3B 3B 3B 3B 3B 3B 3B:D0
4230- 3B 3B 3B 3B 3B C0 D5 AA:66
4238- D5 AA D5 AA D5 AA D5 AA:FC
4240- D5 81 44 6E 6E 6E 6E 6E:C0
4248- 6E 6E 6E 6E 6E 6E 6E 66:68
4250- 23 EE 28 11 2A 55 2A EE:E1
4258- 28 11 2A 55 2A 15 B7 04:B2
4260- 80 80 80 80 80 80 80 10:90
4268- B7 54 08 55 2A 55 2A EE:FF
4270- 28 11 2A 55 2A EE 28 61:59
4278- 00 00 00 00 00 00 00 00:00
-----------------------------------
SUM  FF 7F 6E 28 22 99 68 28 :5F

P:5  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4280- 83 3C 00 00 1E 00 00 3C:19
4288- 00 00 00 00 00 00 40 07:47
4290- 70 01 3C 70 01 00 60 03:81
4298- 00 3C 1E 00 60 73 01 00:2E
42A0- 60 03 40 3F 1E 00 00 E0:E0
42A8- 33 3B 3B 3B 3B 3B 3B 3B:D0
42B0- 3B 3B 3B 3B BB BB 96 CC:C4
42B8- 99 B3 E6 9C 99 B3 E6 8C:8C
42C0- 96 44 44 44 6E 6E 6E 6E:1A
42C8- 6E 6E 6E 6E 6E 6E 6E E6:E8
42D0- 23 55 2A 55 2A 55 2A 55:F5
42D8- 2A 04 22 44 28 55 72 FF:82
42E0- D5 AA D5 AA D5 AA D5 FF:51
42E8- 27 55 0A 11 22 10 2A 55:48
42F0- 2A 55 2A 55 2A 55 2A 65:0C
42F8- 00 00 00 00 00 00 00 00:00
4300- 83 7C 03 00 7E 01 00 7C:FD
4308- 03 00 00 00 00 00 40 3F:82
4310- 7E 01 60 73 0F 00 60 1F:E0
4318- 00 00 7E 01 60 73 0F 00:61
-----------------------------------
SUM  D5 81 DE 90 68 25 A8 F4 :ED

P:6  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4320- 60 03 00 3C 7E 01 00 E0:FE
4328- B3 BB BB BB BB BB BB BB:D0
4330- BB BB BB BB BB BB B7 FE:17
4338- 99 FB A9 D1 CA EF FF BF:85
4340- B7 44 44 44 44 6E 6E 6E:11
4348- 6E 6E 6E 6E 6E 6E 6E E6:E8
4350- 03 01 02 04 08 10 20 40:82
4358- 28 14 2A 55 2A 80 80 80:65
4360- 80 80 80 80 80 80 80 80:00
4368- 80 80 2A 55 2A 14 0A 05:CC
4370- 08 10 20 40 00 01 02 64:DF
4378- 00 00 00 00 00 00 00 00:00
4380- 83 7C 7F 3F 7E 7F 7F 7C:B5
```

```
4388- 7F 7F 01 00 00 7F 7F 3F:3C
4390- 7E 01 60 73 7F 7F 67 7F:36
4398- 7F 3F 7E 01 60 73 7F 7F:0E
43A0- 67 7F 7F 3F 7E 7F 1F E0:A0
43A8- A3 D5 AA D5 AA D5 AA D5:F5
43B0- AA D5 AA D5 AA 95 B7 FE:F2
43B8- 99 BB BA DD AE EE FF BF:45
---------------------------------
SUM  0B 6A B2 1C 29 2E DC 80 :F6

P:7  +0 +1 +2 +3 +4 +5 +6 +7 SUM
43C0- B7 D5 AA D5 AA D5 AA D5:09
43C8- AA D5 AA D5 AA D5 AA E5:0C
43D0- 23 45 0A 15 2A 54 28 51:7E
43D8- 2A 45 7F 7F D7 AA D5 AA:6D
43E0- D5 AA D5 AA D5 AA D5 AA:FC
43E8- D5 AA FD 7F 7F 51 2A 15:0A
43F0- 2A 54 28 51 22 45 0A 65:CD
43F8- 00 00 00 00 00 00 00 00:00
4400- FF FF FF FF FF FF FF FF:F8
4408- FF FF FF FF FF FF FF FF:F8
4410- FF FF FF FF FF FF FF FF:F8
4418- FF FF FF FF FF FF FF FF:F8
4420- FF FF FF FF FF FF FF FF:F8
4428- 87 00 00 00 00 00 00 00:87
4430- 00 00 00 00 00 00 00 00:00
4438- 00 00 00 00 00 00 00 00:00
4440- 00 00 00 00 00 00 00 00:00
4448- 00 00 00 00 00 00 00 70:70
4450- F3 81 FE BF 82 C0 E2 81:D6
4458- FE BF 82 C0 E2 BF 93 FE:31
---------------------------------
SUM  F5 17 52 32 2A 62 CA C3 :A9

P:8  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4460- E6 BB AA D5 AA EE FF BF:76
4468- 93 FE EF 84 00 C7 FF 81:4B
4470- FE BF 82 C0 E2 81 FE E7:47
4478- 00 00 00 00 00 00 00 00:00
4480- 83 00 00 00 00 0C 78 0E:15
4488- 60 40 01 07 03 0F 78 39:6B
4490- 00 00 00 60 01 43 39 30:0D
4498- 00 03 07 1C 18 00 18 0F:65
44A0- 18 00 00 00 00 00 00 E0:F8
44A8- D3 AA D5 AA D5 AA D5 AA:FA
44B0- D5 AA D5 AA D5 AA D5 FF:51
44B8- FF FF FF FF FF FF FF FF:F8
44C0- D5 AA D5 AA D5 AA D5 AA:FC
44C8- D5 AA D5 AA D5 AA D5 E2:34
44D0- F3 A6 FC BF 82 C0 E2 A6:1E
44D8- FC BF 82 C0 E2 BF 93 FE:2F
44E0- E6 A3 AA D5 AA E2 FF BF:52
44E8- 93 FE EF 84 80 C7 FF A6:F0
44F0- FC BF 82 C0 E2 A6 FC E7:68
44F8- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  27 27 0F DB 6B 09 FF B1 :5C

P:9  +0 +1 +2 +3 +4 +5 +6 +7 SUM
4500- 83 00 00 00 00 0C 18 40:E7
4508- 03 07 07 1F 60 0D 18 70:25
4510- 00 00 00 00 0C 18 1B 30:6F
4518- 00 03 06 0C 18 00 00 0C:39
4520- 18 67 00 00 00 00 00 E0:5F
4528- D3 AA D5 D5 D2 AA D5 AA:22
4530- D5 D5 D2 AA E5 FF FF FF:08
4538- FF FF FF FF FF FF FF FF:F8
4540- FF FF D3 AA D5 D5 D2 AA:A1
4548- D5 AA D5 D5 D4 AA D5 E2:5E
4550- F3 A6 FC BF FF FF CF A6:C7
4558- FC BF FF FF CF FF C0 FF:46
4560- E6 83 80 80 80 E0 FF FF:C7
4568- C0 FF EF FF FF 9F FF A6:F0
4570- FC BF FF FF CF A6 FC E7:11
4578- 00 00 00 00 00 00 00 00:00
4580- 83 00 00 00 00 0E 78 01:0A
4588- 00 40 07 7F 60 0F 38 0E:7B
4590- 38 00 00 00 60 30 40 03:0B
4598- 3F 70 07 7C 41 3F 70 0E:30
---------------------------------
SUM  A4 EE D2 5F 00 07 AE 51 :C9

P:10 +0 +1 +2 +3 +4 +5 +6 +7 SUM
45A0- 78 7F 63 00 00 00 00 E0:3A
45A8- D3 5D 5D D9 5D 5D 5D 5D:DA
45B0- 5D D9 5D 5D E5 FB F7 FA:C1
45B8- F7 BB F7 FA F7 BB F7 FA:46
45C0- F7 EF D3 5D 5D D1 5D 5D:FE
45C8- 5D 5D 5D D1 5D 5D 5D E5:E4
45D0- F3 A6 FC BF E6 FF FF A6:DE
```

```
4700- 83 7C 03 00 7E 01 00 7C:FD
4708- 03 00 00 00 00 00 40 3F:82
4710- 7E 01 60 73 0F 00 60 1F:E0
4718- 00 00 7E 01 60 73 0F 00:61
4720- 60 03 00 3C 7E 01 00 E0:FE
4728- D3 DD 5D DD 5D DD 5D DD:5E
4730- 5D DD 5D DD 5D DD 93 FE:3F
4738- E7 FB A8 D9 8A EF FF BF:9A
4740- 93 09 09 09 09 5D 5D 5D:CE
4748- 5D 5D 5D 5D 5D 5D 5D E5:70
4750- 23 45 0A 15 2A 54 28 51:7E
4758- 28 54 2A 55 4A FF D7 AA:C5
4760- D5 AA D5 AA D5 AA D5 AA:FC
4768- FD FF 29 55 2A 15 0A 15:D8
4770- 2A 54 28 51 22 45 0A 65:CD
4778- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  5C 5A B7 3C 5C 08 14 EF :10

P:13 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4780- 83 7C 7F 3F 7E 7F 7F 7C:B5
4788- 7F 7F 01 00 00 7F 7F 3F:3C
4790- 7E 01 60 73 7F 7F 67 7F:36
4798- 7F 3F 7E 01 60 73 7F 7F:0E
47A0- 67 7F 7F 3F 7E 7F 1F E0:A0
47A8- 83 80 80 80 80 80 80 80:03
47B0- 80 80 80 80 80 80 93 FE:91
47B8- E6 BB AA D5 AA EE FF BF:76
47C0- 93 80 80 80 80 80 80 80:13
47C8- 80 80 80 80 80 80 80 E0:60
47D0- 23 55 2A 55 2A 55 2A 55:F5
47D8- 2A 78 7F FF D5 AA D5 AA:1E
47E0- D5 AA D5 AA D5 AA D5 AA:FC
47E8- D5 AA F5 FF 7F 0F 2A 55:80
47F0- 2A 55 2A 55 2A 55 2A 65:0C
47F8- 00 00 00 00 00 00 00 00:00
4800- BF 80 80 80 80 F0 99 9E:E6
4808- 80 9E BC 80 BC 83 E0 8C:05
4810- 98 80 80 8C 80 C3 81 9F:87
4818- 80 DE 81 B0 8F C0 BF 83:20
---------------------------------
SUM  DA 67 61 55 4D 60 F6 E5 :7F

P:14 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4820- 80 80 E0 80 80 80 80 FE:DE
```

```
45D8- FC BF E6 FF FF BF 93 FE:EF
45E0- 86 BF D5 AA D5 FE F9 BF:4F
45E8- 93 FE EF CC FF FF FF A6:EF
45F0- FC BF E6 FF FF A6 FC E7:28
45F8- 00 00 00 00 00 00 00 00:00
4600- 83 00 00 00 00 00 00 00:83
4608- 00 00 00 00 00 00 00 00:00
4610- 00 00 00 00 00 00 00 00:00
4618- 00 00 00 00 00 00 00 00:00
4620- 00 00 00 00 00 00 00 E0:E0
4628- D3 5D 5D 5D 5D 5D 5D 5D:5E
4630- 5D 5D 5D 5D 5D 14 AA D5:64
4638- AA D5 AA D5 AA D5 AA D5:FC
---------------------------------
SUM  54 2C 34 20 0F E8 3C 4A :51

P:11 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4640- AA 94 09 5D 5D 5D 5D 5D:18
4648- 5D 5D 5D 5D 5D 5D 5D E5:70
4650- 23 A6 28 44 28 55 2A A6:82
4658- 28 44 2A 55 2A 15 93 64:21
4660- DF AA D5 AA D5 AA FD 13:97
4668- 93 14 22 55 2A 55 2A A6:6D
4670- 28 44 28 55 2A A6 28 64:45
4678- 00 00 00 00 00 00 00 00:00
4680- 83 3C 00 00 1E 00 00 3C:19
4688- 00 00 00 00 00 00 40 07:47
4690- 70 01 3C 70 01 00 60 03:81
4698- 00 18 1E 00 60 73 01 00:0A
46A0- 60 03 00 3F 1E 00 00 E0:A0
46A8- D3 5D 5D 5D 5D 5D 5D 5D:5E
46B0- 5D 5D 5D 5D 5D 9D C0 B3:E1
46B8- E6 CC D9 9E E6 CC 99 B3:27
46C0- C0 09 09 09 5D 5D 5D 5D:4F
46C8- 5D 5D 5D 5D 5D 5D 5D E5:70
46D0- 23 55 2A 55 2A 55 2A 55:F5
46D8- 2A 10 08 10 28 15 00 80:0F
---------------------------------
SUM  BF 86 5C 79 7E 26 01 69 :28

P:12 +0 +1 +2 +3 +4 +5 +6 +7 SUM
46E0- 80 80 80 80 80 80 80 80:00
46E8- 00 54 0A 04 08 04 2A 55:ED
46F0- 2A 55 2A 55 2A 55 2A 65:0C
46F8- 00 00 00 00 00 00 00 00:00
```

```
4CF0- FC BF 82 C0 E2 A6 FC E7:68
4CF8- 00 00 00 00 00 00 00 00:00
4D00- 83 00 00 00 00 0C 18 70:17
4D08- 70 40 43 71 40 0F 78 40:6B
4D10- 03 00 00 00 1C 40 1B 30:AA
4D18- 00 03 06 0C 18 00 00 0C:39
----------------------------------
SUM  D8 9D 7E 6E B5 60 49 3C :FB

P:22 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4D20- 18 54 00 00 00 00 00 E0:4C
4D28- D3 AA D5 85 D5 AA D5 AA:D5
4D30- D5 95 D5 AA E5 FB FF FB:C3
4D38- FF FF FF FB FF FF FF FF:F4
4D40- FF EF D3 AA D5 95 D5 AA:54
4D48- D5 AA D5 95 D5 AA D5 E2:1F
4D50- F3 A6 FC BF E6 FF FF A6:DE
4D58- FC BF E6 FF FF BF 93 FE:EF
4D60- E6 F3 D7 AA F5 E7 FF BF:F4
4D68- 93 FE EF CC FF FF FF A6:EF
4D70- FC BF E6 FF FF A6 FC E7:28
4D78- 00 00 00 00 00 00 00 00:00
4D80- 83 00 00 00 00 03 60 00:E6
4D88- 00 40 3D 00 3C 03 60 03:1F
4D90- 60 00 00 00 40 19 00 1E:D7
4D98- 00 5E 01 30 0F 40 0F 03:F0
4DA0- 00 00 60 00 00 00 00 E0:40
4DA8- 83 80 40 81 80 80 80 80:C4
4DB0- 40 81 80 80 E0 FB F7 EA:7D
4DB8- FF EB F7 EA FF EB F7 EA:96
----------------------------------
SUM  9C CA 34 B7 25 F2 46 58 :06

P:23 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4DC0- FF EF 83 00 40 81 00 00:32
4DC8- 00 00 40 81 00 00 00 E0:A1
4DD0- F3 A6 FC BF E6 FF FF A6:DE
4DD8- FC BF E6 FF FF BF 93 FE:EF
4DE0- F3 AF D5 AA D5 FA E7 BF:96
4DE8- 93 FE EF CC FF FF FF A6:EF
4DF0- FC BF E6 FF FF A6 FC E7:28
4DF8- 00 00 00 00 00 00 00 00:00
4E00- 83 3C 00 00 7E 7F 7F 7C:B7
4E08- 7F 7F 01 00 00 00 40 07:46
4E10- 70 7F 3F 70 7F 7F 67 7F:82
4E18- 7F 3F 1E 00 60 73 7F 7F:AD
4E20- 67 7F 7F 3F 1E 00 00 E0:A2
4E28- 83 80 80 80 80 80 80 80:03
4E30- 80 80 80 80 80 C0 BF 80:7F
4E38- 80 80 80 80 80 80 80 C0:40
4E40- BF 80 80 00 00 00 00 00:BF
4E48- 00 00 00 00 00 00 00 E0:E0
4E50- 23 A6 20 44 2A 55 2A A6:7C
4E58- 20 44 2A 55 2A 15 93 80:35
----------------------------------
SUM  4D A2 76 7C 47 79 95 F7 :2D

P:24 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4E60- 80 80 80 80 80 80 80 80:00
4E68- 93 10 22 55 2A 55 2A A6:69
4E70- 20 44 2A 55 2A A6 20 64:37
4E78- 00 00 00 00 00 00 00 00:00
4E80- 83 3C 00 00 1E 00 00 3C:19
4E88- 00 00 00 00 00 00 40 07:47
4E90- 70 01 3C 70 01 00 60 03:81
4E98- 00 00 1E 00 60 73 01 00:F2
4EA0- 60 03 00 3C 1E 00 00 E0:9D
4EA8- 83 80 80 80 80 80 80 80:03
4EB0- 80 80 80 80 80 80 93 BE:51
4EB8- E6 CC F5 FF F5 CC 99 B3:B3
4EC0- 93 80 80 80 00 00 00 00:13
4EC8- 00 00 00 00 00 00 00 E0:E0
4ED0- 23 55 2A 55 2A 55 2A 55:F5
4ED8- 08 10 2A 54 28 15 FF AF:81
4EE0- D5 AA D5 AA D5 AA D5 FE:50
4EE8- BF 54 0A 15 2A 04 08 55:BD
4EF0- 2A 55 2A 55 2A 55 2A 65:0C
4EF8- 00 00 00 00 00 00 00 00:00
----------------------------------
SUM  EB 18 F8 12 E1 27 47 3D :99

P:25 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4F00- 83 7C 03 00 7E 01 00 7C:FD
4F08- 03 00 00 00 00 06 40 3F:88
4F10- 7E 01 60 73 0F 00 60 1F:E0
4F18- 00 18 7E 01 60 73 0F 00:79
4F20- 60 03 00 3C 7E 01 00 E0:FE
4F28- 83 80 00 80 00 80 00 80:83
4F30- 00 80 00 80 00 80 93 FE:11
4F38- E6 A3 AA D5 AA E2 FF BF:52
```

```
-----------------------------------
SUM  6C 86 E6 7B 15 8D 16 DF :EA

P:18 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4AA0- 60 03 00 3C 1E 00 00 E0:9D
4AA8- E3 EE 6E 6E 6E 6E 6E 6E:65
4AB0- 6E 6E 6E 6E 6E 2E 96 CE:B8
4AB8- 99 B3 D6 8C 9D B3 E6 8C:70
4AC0- 96 12 12 12 3B 3B 3B 3B:B8
4AC8- 3B 3B 3B 3B 3B 3B 3B E3:80
4AD0- 23 55 2A 55 2A 55 2A 55:F5
4AD8- 2A 04 2A 54 28 15 FF AF:97
4AE0- D5 AA D5 AA D5 AA D5 FE:50
4AE8- BF 54 0A 15 2A 10 2A 55:EB
4AF0- 2A 55 2A 55 2A 55 2A 65:0C
4AF8- 00 00 00 00 00 00 00 00:00
4B00- 83 7C 03 00 7E 01 00 7C:FD
4B08- 03 00 00 00 00 00 40 3F:82
4B10- 7E 01 60 73 0F 00 60 1F:E0
4B18- 00 00 7E 01 60 73 0F 00:61
4B20- 60 03 00 3C 7E 01 00 E0:FE
4B28- A3 AE 2E AE 2E AE 2E AE:E5
4B30- 2E AE 2E AE 2E AE B7 FE:49
4B38- F9 BB AA DD AA EE FF BF:91
-----------------------------------
SUM  54 A2 43 97 F9 FD 45 A7 :B2

P:19 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4B40- B7 12 12 12 12 3B 3B 3B:B0
4B48- 3B 3B 3B 3B 3B 3B 3B E3:80
4B50- 23 45 0A 15 2A 54 28 51:7E
4B58- 28 55 2A 55 70 FF D5 AA:EA
4B60- D5 AA D5 AA D5 AA D5 AA:FC
4B68- F5 FF 07 55 2A 15 0A 15:AE
4B70- 2A 54 28 51 22 45 0A 65:CD
4B78- 00 00 00 00 00 00 00 00:00
4B80- 83 7C 7F 1F 7C 7F 3F 78:4F
4B88- 7F 7F 00 00 00 7E 7F 1F:1A
4B90- 7C 00 40 61 7F 7F 43 7F:DD
4B98- 7F 1F 7C 00 40 61 7F 7F:B9
4BA0- 43 7F 7F 1F 7E 7F 0F E0:4C
4BA8- 83 FE 80 80 80 80 80 FE:FF
4BB0- 80 80 80 80 80 80 B7 FE:B5
4BB8- 99 BB AA D5 AA EE FF BF:29
4BC0- B7 80 80 80 80 80 80 FE:B5
4BC8- 80 80 80 80 80 FE 80 E0:DE
4BD0- 23 55 2A 55 2A 55 2A 55:F5
4BD8- 42 7F 7F BF D5 AA D5 AA:FD
-----------------------------------
SUM  A9 8A 92 8F 6A 94 20 4A :BC

P:20 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4BE0- D5 AA D5 AA D5 AA D5 AA:FC
4BE8- D5 AA D5 FF 7F 7F 21 55:C7
4BF0- 2A 55 2A 55 2A 55 2A 65:0C
4BF8- 00 00 00 00 00 00 00 00:00
4C00- 8F 00 00 00 00 3C 18 07:EA
4C08- 00 38 0E 00 70 07 70 1C:49
4C10- 1C 00 00 1C 40 43 43 07:05
4C18- 00 78 03 78 03 00 1C 07:19
4C20- 00 00 60 00 00 00 00 F8:58
4C28- BF 00 00 00 00 00 00 00:BF
4C30- 00 00 00 00 00 00 00 00:00
4C38- 00 00 00 00 00 00 00 00:00
4C40- 00 00 00 00 00 00 00 00:00
4C48- 00 00 00 00 00 00 00 7E:7E
4C50- F3 A6 FC BF 82 C0 E2 A6:1E
4C58- FC BF 82 C0 E2 BF 93 FE:2F
4C60- E6 BB AA D5 AA EE FF BF:76
4C68- 93 FE EF 84 00 C7 FF A6:70
4C70- FC BF 82 C0 E2 A6 FC E7:68
4C78- 00 00 00 00 00 00 00 00:00
-----------------------------------
SUM  A2 36 DE 2A 21 DE 76 FB :50

P:21 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4C80- 83 00 00 00 00 0C 18 00:A7
4C88- 70 60 03 5C 43 0D 38 0E:C5
4C90- 00 00 00 40 03 66 1C 30:F5
4C98- 00 03 06 0C 18 00 50 0E:8B
4CA0- 18 6E 00 00 00 00 00 E0:66
4CA8- D3 AA F5 AF D5 AA D5 AA:1F
4CB0- F5 AF D5 AA D5 AA F5 FF:96
4CB8- FF FF FF FF FF FF FF FF:F8
4CC0- D7 AA D5 AA F5 AF D5 AA:23
4CC8- D5 AA F5 AF D5 AA D5 E2:59
4CD0- F3 A6 FC BF 82 C0 E2 A6:1E
4CD8- FC BF 82 C0 E2 BF 93 FE:2F
4CE0- E6 BB AA D5 AA EE FF BF:76
4CE8- 93 FE EF 84 80 C7 FF A6:F0
```

```
4828- 8F 00 00 00 00 00 00 00:8F
4830- 00 00 00 00 00 00 00 00:00
4838- 00 00 00 00 00 00 00 00:00
4840- 00 00 00 00 00 00 00 00:00
4848- 00 00 00 00 00 00 00 78:78
4850- F3 AC FC CF 88 C0 E3 AC:41
4858- FC CF 88 C0 E3 BF B7 FE:6A
4860- 99 A3 AA D5 AA E2 FF BF:05
4868- B7 FE 9F 91 00 C5 FF AC:55
4870- FC CF 88 C0 E3 AC FC E7:85
4878- 00 00 00 00 00 00 00 00:00
4880- 83 00 00 00 00 0C 38 05:CC
4888- 60 40 03 0E 43 0F 78 1C:97
4890- 00 00 00 40 01 67 19 30:F1
4898- 00 03 06 0C 18 00 38 0F:74
48A0- 18 54 00 00 00 00 00 E0:4C
48A8- D3 AA D5 AB D5 AA D5 AA:FB
48B0- D5 AB D5 AA D5 AA F5 FF:72
48B8- FF FF FF FF FF FF FF FF:F8
------------------------------------
SUM  EC 56 E7 E3 7D 27 DE 5A :E8

P:15 +0 +1 +2 +3 +4 +5 +6 +7 SUM
48C0- D7 AA D5 AA D5 AB D5 AA:FF
48C8- D5 AA D5 AB D5 AA D5 E2:35
48D0- F3 EE FC CF 88 C0 E3 EE:C5
48D8- FC CF 88 C0 E3 BF B7 FE:6A
48E0- 99 BB AA D5 AA EE FF BF:29
48E8- B7 FE 9F 91 80 C5 FF EE:17
48F0- FC CF 88 C0 E3 EE FC E7:C7
48F8- 00 00 00 00 00 00 00 00:00
4900- 83 00 00 00 00 0C 18 60:07
4908- 61 03 43 3B 40 0D 38 60:C7
4910- 01 00 00 00 0C 18 1B 30:70
4918- 00 03 06 0C 18 00 00 0C:39
4920- 18 6E 00 00 00 00 00 E0:66
4928- D3 AA D5 D5 D4 AA D5 AA:24
4930- D5 D5 D4 AA E5 AF DF BE:59
4938- DF AA DF BE DF AA DF AA:38
4940- DF FA D3 AA D5 95 D4 AA:3E
4948- D5 AA D5 D5 D4 AA D5 E2:5E
4950- F3 EE FC CF 80 80 C0 EE:5A
4958- FC CF 80 80 C0 BF 96 FE:DE
------------------------------------
SUM  0E 97 F4 5C 07 27 3B 72 :D0

P:16 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4960- 99 F3 D7 AA F5 E7 FF BF:A7
4968- 96 FE 9F 80 80 80 FF EE:A0
4970- FC CF 80 80 C0 EE FC E7:5C
4978- 00 00 00 00 00 00 00 00:00
4980- 83 00 00 00 00 07 70 00:FA
4988- 00 40 0F 00 70 07 70 07:3D
4990- 70 00 00 00 60 39 00 07:10
4998- 00 78 03 78 03 00 3C 07:39
49A0- 00 00 60 00 00 00 00 E0:40
49A8- E3 EE 6E E9 6E 6E 6E 6E:E0
49B0- 6E E9 6E 6E E6 FB FD FB:0C
49B8- DF FE FD FB DF FE FD FB:AA
49C0- DF EF 33 3B 7B B9 3B 3B:E6
49C8- 3B 3B 7B B9 3B 3B BB E3:BE
49D0- F3 EE FC CF F9 FF FF EE:91
49D8- FC CF F9 FF FF BF B7 FE:36
49E0- 81 80 80 80 80 80 E0 BF:A0
49E8- B7 FE 9F F3 FF FF FF EE:32
49F0- FC CF F9 FF FF EE FC E7:93
49F8- 00 00 00 00 00 00 00 00:00
------------------------------------
SUM  8B 81 FC A8 67 22 05 8B :C9

P:17 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4A00- 83 18 00 00 7C 7F 3F 78:4D
4A08- 7F 7F 00 00 00 00 00 03:01
4A10- 60 7F 1F 60 7F 7F 43 7F:1E
4A18- 7F 1F 0C 00 40 61 7F 7F:49
4A20- 43 7F 7F 1F 0C 00 00 E0:4C
4A28- E3 EE 6E 6E 6E 6E 6E 6E:65
4A30- 6E 6E 6E 6E 6E 00 80 80:26
4A38- 80 80 80 80 80 80 80 80:00
4A40- 80 80 12 3B 3B 3B 3B 3B:39
4A48- 3B 3B 3B 3B 3B 3B BB E3:00
4A50- 23 EE 08 11 2A 55 2A EE:C1
4A58- 08 11 2A 55 2A 15 B7 64:F2
4A60- DF AA D5 AA D5 AA FD 13:97
4A68- B7 44 08 55 2A 55 2A EE:EF
4A70- 08 11 2A 55 2A EE 08 61:19
4A78- 00 00 00 00 00 00 00 00:00
4A80- 83 3C 00 00 1E 00 00 3C:19
4A88- 00 00 00 00 00 00 40 07:47
4A90- 70 01 3C 70 01 00 60 03:81
4A98- 00 00 1E 00 60 73 01 00:F2
```

```
4F40- 93 80 80 80 80 80 80 80:13
4F48- 80 80 80 80 80 80 80 E0:60
4F50- 23 45 0A 15 2A 54 28 51:7E
4F58- 28 55 2A 15 7F BF D5 AA:79
4F60- D5 AA D5 AA D5 AA D5 AA:FC
4F68- D5 FF 7F 54 2A 55 0A 15:45
4F70- 2A 54 28 51 22 45 0A 65:CD
4F78- 00 00 00 00 00 00 00 00:00
4F80- 83 00 00 00 00 00 00 00:83
4F88- 00 00 00 00 00 00 00 00:00
4F90- 00 00 00 00 00 00 00 00:00
4F98- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  82 D2 3B FE DF B4 27 76 :BD

P:26 +0 +1 +2 +3 +4 +5 +6 +7 SUM
4FA0- 00 00 00 00 00 00 00 E0:E0
4FA8- B3 80 98 B3 E6 CC 99 80:49
4FB0- 98 B3 E6 CC 99 B3 93 FE:DA
4FB8- E6 A3 AA D5 AA E2 FF BF:52
4FC0- 93 B2 E6 CC 99 B3 E6 80:A9
4FC8- 98 B3 E6 CC 99 80 E4 E6:E0
4FD0- 23 55 2A 55 2A 55 2A 55:F5
4FD8- 7C 7F 7F AF D5 AA D5 AA:27
4FE0- D5 AA D5 AA D5 AA D5 AA:FC
4FE8- D5 AA D5 FE 7F 7F 1F 55:C4
4FF0- 2A 55 2A 55 2A 55 2A 65:0C
4FF8- 00 00 00 00 00 00 00 00:00
5000- 87 00 00 00 00 0F 58 03:F1
5008- 00 70 07 7E 61 0F 38 38:D5
5010- 0E 00 00 38 60 01 67 03:11
5018- 3F 70 07 7C 41 3F 08 0F:C9
5020- 78 7F 63 00 00 00 00 F0:4A
5028- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5030- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5038- 7F 7F 7F 7F 7F 7F 7F 7F:F8
---------------------------------
SUM  98 94 5F 9C 57 EC 8E A0 :98

P:27 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5040- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5048- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5050- F3 EE FC CF 88 C0 E3 EE:C5
5058- FC CF 88 C0 E3 BF B7 FE:6A
5060- 99 BB AA D5 AA EE FF BF:29
5068- B7 FE 9F 91 00 C5 FF EE:97
5070- FC CF 88 C0 E3 EE FC E7:C7
5078- 00 00 00 00 00 00 00 00:00
5080- 83 00 00 00 00 0C 18 00:A7
5088- 30 60 07 78 61 0D 18 06:9B
5090- 00 00 00 00 03 66 1C 30:B5
5098- 00 03 06 0C 18 00 00 0C:39
50A0- 18 67 00 00 00 00 00 E0:5F
50A8- D3 AA D5 A9 D5 AA D5 AA:F9
50B0- D5 A9 D5 AA D5 AA F5 FF:70
50B8- FF FF FF FF FF FF FF FF:F8
50C0- D7 AA D5 AA D5 A9 D5 AA:FD
50C8- D5 AA D5 A9 D5 AA D5 E2:33
50D0- F3 EE FC CF F8 9F E3 EE:14
50D8- FC CF F8 9F E3 BF B7 FE:B9
---------------------------------
SUM  46 70 A7 4A A0 A1 EB C0 :93

P:28 +0 +1 +2 +3 +4 +5 +6 +7 SUM
50E0- 99 BB 80 80 80 EE FF BF:80
50E8- B7 FE 9F F1 FF C4 FF EE:F5
50F0- FC CF F8 9F E3 EE FC E7:16
50F8- 00 00 00 00 00 00 00 00:00
5100- 83 00 00 00 00 0C 18 38:DF
5108- 38 68 41 61 01 0F 78 01:CB
5110- 03 00 00 00 18 40 39 30:C4
5118- 00 03 07 1C 18 00 50 0E:9C
5120- 18 00 00 00 00 00 00 E0:F8
5128- D3 AA D5 A9 D5 AA D5 AA:F9
5130- D5 A5 D5 AA E5 FB DD FB:B1
5138- DD FF DD FB DD FF DD FF:6C
5140- DD EF D3 AA D5 A5 D5 AA:42
5148- D5 AA D5 A5 D5 AA D5 E2:2F
5150- F3 EE FC CF F9 FF FF EE:91
5158- FC CF F9 FF FF BF B7 FE:36
5160- 99 80 80 80 80 80 FF BF:D7
5168- B7 FE 9F F3 FF FF FF EE:32
5170- FC CF F9 FF FF EE FC E7:93
5178- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  94 E4 9B 6A 4A 19 FC 9B :77

P:29 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5180- 83 00 00 00 00 7F 7F 7F:00
5188- 7F 7F 71 7F 0F 7F 7F 7F:7A
```

```
5190- 7F 00 00 00 40 1F 00 78:56
5198- 7F 47 7F 3F 7C 7F 01 7F:FF
51A0- 7F 7F 7F 00 00 00 00 E0:5D
51A8- D3 D8 58 59 58 58 58 58:BC
51B0- 58 59 58 58 E0 FB FF FB:36
51B8- F7 FB FF FB F7 FB FF FB:D8
51C0- F7 EF 03 0D 4D 09 0D 0D:66
51C8- 0D 0D 4D 09 0D 0D 8D E7:FE
51D0- 83 EE 80 CC 89 80 80 EE:34
51D8- 80 CC 89 80 80 80 B7 80:8C
51E0- F0 AF D5 AA D5 FA 87 80:F4
51E8- B7 80 98 93 80 80 80 EE:D0
51F0- 80 CC 89 80 80 EE 80 E0:23
51F8- 00 00 00 00 00 00 00 00:00
5200- 83 3C 00 00 7E 7F 7F 7C:B7
5208- 7F 7F 01 00 00 00 40 07:46
5210- 70 7F 3F 70 7F 7F 67 7F:82
5218- 7F 3F 1E 00 60 73 7F 7F:AD
----------------------------------
SUM  C0 9B CB F9 8F D9 52 54 :2D

P:30 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5220- 67 7F 7F 3F 1E 00 00 E0:A2
5228- D3 D8 58 58 58 58 58 58:BB
5230- 58 58 58 58 58 18 80 CC:1C
5238- 99 B3 E6 CC 99 B3 E6 8C:BC
5240- 80 49 0D 0D 0D 0D 0D 0D:17
5248- 0D 0D 0D 0D 0D 0D 8D E7:C2
5250- 23 80 08 51 2A 55 2A 80:25
5258- 08 51 2A 55 2A 15 80 FF:96
5260- D7 AA D5 AA D5 AA F5 9F:13
5268- 80 44 28 55 2A 55 2A 80:6A
5270- 08 51 2A 55 2A 80 08 61:EB
5278- 00 00 00 00 00 00 00 00:00
5280- 83 7C 01 00 7E 7F 3F 7C:B8
5288- 7F 7F 00 00 00 00 40 0F:4D
5290- 7C 7F 7F 71 7F 7F 63 0F:5B
5298- 00 00 7E 7F 7F 73 7F 7F:ED
52A0- 63 03 00 3C 7E 00 00 E0:00
52A8- D3 D8 58 58 58 58 58 58:BB
52B0- 58 58 58 58 58 18 B7 FE:B5
52B8- 99 FF FD C1 D7 FF FF BF:EA
----------------------------------
SUM  E7 74 33 6C 7F 06 98 91 :A8

P:31 +0 +1 +2 +3 +4 +5 +6 +7 SUM
52C0- B7 0D 0D 0D 0D 0D 0D 0D:12
52C8- 0D 0D 0D 0D 0D 0D 0D E7:42
52D0- 23 55 2A 55 2A 55 2A 55:F5
52D8- 20 14 2A 54 28 01 80 80:DB
52E0- 80 80 80 80 80 80 80 80:00
52E8- 80 40 0A 15 2A 14 02 55:74
52F0- 2A 55 2A 55 2A 55 2A 65:0C
52F8- 00 00 00 00 00 00 00 00:00
5300- 83 7C 03 00 7E 01 00 7C:FD
5308- 03 00 00 00 00 0F 40 3F:91
5310- 7E 01 60 73 0F 00 60 1F:E0
5318- 00 3C 7E 01 60 73 0F 00:9D
5320- 60 03 00 3C 7E 01 00 E0:FE
5328- D3 D8 58 58 58 58 58 58:BB
5330- 58 58 58 58 58 58 B7 FE:C5
5338- 99 BB AA D5 AA EE FF BF:29
5340- B7 0D 0D 0D 0D 0D 0D 0D:12
5348- 0D 0D 0D 0D 0D 0D 0D E7:42
5350- 23 45 0A 15 2A 54 28 51:7E
5358- 28 55 2A 61 7F AF D5 AA:B5
----------------------------------
SUM  68 F3 AB 72 C8 98 44 C1 :DD

P:32 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5360- D5 AA D5 AA D5 AA D5 AA:FC
5368- D5 FE 7F 43 2A 55 0A 15:33
5370- 2A 54 28 51 22 45 0A 65:CD
5378- 00 00 00 00 00 00 00 00:00
5380- 83 00 00 00 00 00 00 00:83
5388- 00 00 00 00 00 00 00 00:00
5390- 00 00 00 00 00 00 00 00:00
5398- 00 00 00 00 00 00 00 00:00
53A0- 00 00 00 00 00 00 00 E0:E0
53A8- C3 AC E4 CC 99 B3 E6 AC:FD
53B0- E4 CC 99 B3 E6 8C B7 FE:23
53B8- 99 BB AA D5 AA EE FF BF:29
53C0- B7 CC 99 B3 E6 CC 99 AC:C6
53C8- E4 CC 99 B3 E6 AC 98 E1:07
53D0- 23 55 2A 55 2A 55 2A 05:A5
53D8- 7F 7F 7F AB D5 AA D5 AA:26
53E0- D5 AA D5 AA D5 AA D5 AA:FC
53E8- D5 AA D5 FA 7F 7F 7F 50:1B
53F0- 2A 55 2A 55 2A 55 2A 65:0C
53F8- 00 00 00 00 00 00 00 00:00
```

```
-----------------------------------
SUM  A8 44 52 F1 93 66 33 08 :63

P:33 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5400- 87 00 00 00 00 03 58 01:E3
5408- 00 60 03 47 43 0D 18 1C:2E
5410- 07 00 00 30 60 01 67 61:60
5418- 73 61 07 7C 70 71 03 0F:4A
5420- 18 00 67 00 00 00 00 F0:6F
5428- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5430- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5438- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5440- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5448- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5450- F3 A6 FC BF 82 C0 E2 A6:1E
5458- FC BF 82 C0 E2 BF 93 FE:2F
5460- E6 BB AA D5 AA EE FF BF:76
5468- 93 FE FF 84 00 C7 FF A6:80
5470- FC BF 82 C0 E2 A6 FC E7:68
5478- 00 00 00 00 00 00 00 00:00
5480- 83 00 00 00 00 0C 18 00:A7
5488- 38 70 0E 60 78 0C 08 0E:B0
5490- 00 00 00 00 03 7E 1C 30:CD
5498- 00 03 06 0C 18 00 00 0C:39
-----------------------------------
SUM  B3 8C A9 72 11 6D 00 32 :0A

P:34 +0 +1 +2 +3 +4 +5 +6 +7 SUM
54A0- 78 63 00 00 00 00 00 E0:BB
54A8- D3 AA D5 91 D5 AA D5 AA:E1
54B0- D5 85 C5 AA D5 AA FD FF:44
54B8- FF FF FF FF FF FF FF FF:F8
54C0- DF AA D5 AA D5 C5 D4 AA:20
54C8- D5 AA D5 85 D5 AA D5 E2:0F
54D0- F3 A6 FC BF AA D5 E2 A6:5B
54D8- FC BF AA D5 E2 BF 93 FE:6C
54E0- E6 BB FF FF FF EE FF BF:4A
54E8- 93 FE EF D4 AA C5 FF A6:68
54F0- FC BF AA D5 E2 A6 FC E7:A5
54F8- 00 00 00 00 00 00 00 00:00
5500- 83 00 00 00 00 0C 18 1C:C3
5508- 1C 5C 41 41 01 0F 58 01:63
5510- 07 00 00 00 38 60 31 30:00
5518- 00 03 07 1C 18 00 38 0F:85
5520- 18 00 00 00 00 00 00 E0:F8
5528- D3 AA D5 A9 D5 AA D5 AA:F9
5530- D5 A9 D5 AA E5 FB D7 EB:9F
5538- D7 EF D7 EB D7 EF D7 EF:14
-----------------------------------
SUM  74 63 4A 40 4C BE 45 C4 :74

P:35 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5540- D7 EF D3 AA D5 A9 D5 AA:40
5548- D5 AA D5 A9 D5 AA D5 E2:33
5550- F3 A6 FC BF E6 FF FF A6:DE
5558- FC BF E6 FF FF BF 93 FE:EF
5560- E6 FC D5 AA D5 9F FF BF:93
5568- 93 FE EF CC FF FF FF A6:EF
5570- FC BF E6 FF FF A6 FC E7:28
5578- 00 00 00 00 00 00 00 00:00
5580- 83 00 00 00 00 00 00 00:83
5588- 00 00 00 00 00 00 00 00:00
5590- 00 00 00 00 00 00 00 00:00
5598- 00 00 00 00 00 00 00 00:00
55A0- 00 00 00 00 00 00 00 E0:E0
55A8- A3 2C 6C 29 2C 2C 2C 2C:14
55B0- 6C 29 2C 2C E4 AF D7 BA:11
55B8- FD FE D7 BA FD FE D7 BA:18
55C0- FD FA 93 1A 5A 19 1A 1A:4B
55C8- 1A 1A 5A 19 1A 1A 1A E2:D7
55D0- F3 A6 FC B3 FE FF FF A6:EA
55D8- FC B3 FE FF FF BF 93 BE:BB
-----------------------------------
SUM  A5 77 8A 7A E0 1F D6 5C :51

P:36 +0 +1 +2 +3 +4 +5 +6 +7 SUM
55E0- 80 80 80 80 80 80 80 BF:3F
55E8- 93 FE E7 FC FF FF FF A6:17
55F0- FC B3 FE FF FF A6 FC E7:34
55F8- 00 00 00 00 00 00 00 00:00
5600- 83 3C 00 00 7E 7F 3F 7C:77
5608- 7F 7F 00 00 00 00 40 07:45
5610- 70 7F 3F 70 7F 7F 63 7F:7E
5618- 7F 3F 1E 00 60 73 7F 7F:AD
5620- 63 7F 7F 3F 1E 00 00 E0:9E
5628- A3 2C 2C 2C 2C 2C 2C 2C:D7
5630- 2C 2C 2C 2C 2C 8C 93 B3:AE
5638- E6 CC 99 B3 E6 CC 99 B3:FC
5640- 93 12 12 1A 1A 1A 1A 1A:39
5648- 1A 1A 1A 1A 1A 1A 1A E2:98
```

```
5650- 03 FF 21 54 2A 55 0A FF:FF
5658- 21 54 2A 55 2A 05 9F FF:C1
5660- D7 AA D5 AA D5 AA F5 9F:13
5668- FC 10 2A 55 2A 55 0A FF:13
5670- 21 54 2A 55 0A FF 21 64:82
5678- 00 00 00 00 00 00 00 00:00
------------------------------------
SUM  DD DA D2 66 C8 A6 31 3B :C9

P:37 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5680- 83 7C 03 00 7E 7F 7F 7C:FA
5688- 7F 7F 01 00 00 00 40 3F:7E
5690- 7E 7F 7F 73 7F 7F 67 1F:73
5698- 00 00 7E 7F 7F 73 7F 7F:ED
56A0- 67 03 00 3C 7E 01 00 E0:05
56A8- A3 2C 2C 2C 2C 2C 2C 2C:D7
56B0- 2C 2C 2C 2C 2C AC 93 FE:19
56B8- E6 BF BF 94 DE FE FF BF:92
56C0- 93 12 12 12 1A 1A 1A 1A:31
56C8- 1A 1A 1A 1A 1A 1A 1A E2:98
56D0- 23 45 0A 15 2A 54 28 11:3E
56D8- 08 14 2A 54 28 79 FF AA:E4
56E0- D5 AA D5 AA D5 AA D5 EA:3C
56E8- FF 4F 0A 15 2A 14 08 14:C7
56F0- 2A 54 28 11 22 45 0A 65:8D
56F8- 00 00 00 00 00 00 00 00:00
5700- 83 7C 03 00 7E 01 00 7C:FD
5708- 03 00 00 00 00 0F 40 3F:91
5710- 7E 01 60 73 0F 00 60 1F:E0
5718- 00 3C 7E 01 60 73 0F 00:9D
------------------------------------
SUM  76 1F 60 F3 C4 CF 54 16 :E5

P:38 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5720- 60 03 00 3C 7E 01 00 E0:FE
5728- A3 AC 2C AC 2C AC 2C AC:D7
5730- 2C AC 2C AC 2C AC 93 FE:19
5738- E6 BB AA D5 AA EE FF BF:76
5740- 93 12 12 12 12 1A 1A 1A:29
5748- 1A 1A 1A 1A 1A 1A 1A E6:9C
5750- 23 45 0A 15 2A 54 28 51:7E
5758- 28 55 2A 7E FF AB D5 AA:4E
5760- D5 AA D5 AA D5 AA D5 AA:FC
5768- D5 FA FF 3F 2A 5E 0A 15:AB
5770- 2A 54 28 51 22 45 0A 65:CD
5778- 00 00 00 00 00 00 00 00:00
5780- 83 00 00 00 00 00 00 00:83
5788- 00 00 00 00 00 00 00 00:00
5790- 00 00 00 00 00 00 00 00:00
5798- 00 00 00 00 00 00 00 00:00
57A0- 00 00 00 00 00 00 00 E0:E0
57A8- B3 A6 98 B3 E6 CC 99 A6:95
57B0- 98 B3 E6 CC 99 B3 93 FE:DA
57B8- E6 BB AA D5 AA EE FF BF:76
------------------------------------
SUM  95 E8 86 B6 1F 2B 03 AB :B1

P:39 +0 +1 +2 +3 +4 +5 +6 +7 SUM
57C0- 93 B2 E6 CC 99 B3 E6 A6:CF
57C8- 98 B3 E6 CC 99 A6 E4 E6:06
57D0- 03 00 00 00 00 00 00 78:7B
57D8- 7F 7F 7F AA D5 AA D5 AA:25
57E0- D5 AA D5 AA D5 AA D5 AA:FC
57E8- D5 AA D5 EA 7F 7F 7F 0F:CA
57F0- 00 00 00 00 00 00 00 60:60
57F8- 00 00 00 00 00 00 00 00:00
5800- 83 00 00 00 00 0F 78 61:6B
5808- 0F 60 03 03 43 0F 18 4E:2D
5810- 03 00 00 70 40 01 73 60:87
5818- 40 41 07 3C 30 00 07 0F:0A
5820- 18 00 6E 00 00 00 00 E0:66
5828- 83 00 00 00 00 00 00 00:83
5830- 00 00 00 00 00 00 00 00:00
5838- 00 00 00 00 00 00 00 00:00
5840- 00 00 00 00 00 00 00 00:00
5848- 00 00 00 00 00 00 00 E0:E0
5850- F3 EE FC FF 88 C0 E3 EE:F5
5858- FC FF 88 C0 E3 BF B7 FE:9A
------------------------------------
SUM  B6 C6 F1 44 79 CA 97 91 :1C

P:40 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5860- 99 BB A2 D1 A8 EE FF BF:1B
5868- B7 FE FF 91 00 C5 FF EE:F7
5870- FC FF 88 C0 E3 EE FC E7:F7
5878- 00 00 00 00 00 00 00 00:00
5880- 83 00 00 00 00 0C 18 00:A7
5888- 1C 38 1C 00 1C 0C 00 1C:B4
5890- 00 00 00 00 07 3C 1E 30:91
5898- 00 03 06 0C 18 00 00 0C:39
```

```
58A0- 00 60 00 00 00 00 00 E0:40
58A8- D3 AA D5 95 D4 AA D5 AA:E4
58B0- D5 95 A8 AA D5 AA FD FF:37
58B8- FF FF FF FF FF FF FF FF:F8
58C0- DF AA D5 AA D5 D5 D4 AA:30
58C8- D5 AA D5 D5 C8 AA D5 E2:52
58D0- F3 EE FC CF 80 80 E0 EE:7A
58D8- FC CF 80 80 E0 BF B7 FE:1F
58E0- 99 8B 80 80 80 E8 FF BF:4A
58E8- B7 FE 9F 80 80 C0 FF EE:01
58F0- FC CF 80 80 E0 EE FC E7:7C
58F8- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  81 FA 8C BA 4B 9C 3B 80 :63

P:41 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5900- 83 00 00 00 00 0C 18 0E:B5
5908- 0F 4E 41 41 01 0F 58 03:4A
5910- 0E 00 00 00 30 60 30 70:3E
5918- 40 03 07 1C 38 00 1C 0F:C9
5920- 18 00 54 00 00 00 00 E0:4C
5928- D3 AA D5 A9 D5 AA D5 AA:F9
5930- D5 A9 D5 AA E5 FB DF FF:BB
5938- DF EB DF FF DF EB DF EB:3C
5940- DF EF D3 AA D5 A9 D5 AA:48
5948- D5 AA D5 A9 D5 AA D5 E2:33
5950- F3 EE FC CF F9 FF FF EE:91
5958- FC CF F9 FF FF BF B7 FE:36
5960- 99 FC D5 AA D5 9F FF BF:46
5968- B7 FE 9F F3 FF FF FF EE:32
5970- FC CF F9 FF FF EE FC E7:93
5978- 00 00 00 00 00 00 00 00:00
5980- 83 00 00 00 00 00 00 00:83
5988- 00 00 00 00 00 00 00 00:00
5990- 00 00 00 00 00 00 00 00:00
5998- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  F1 AE 2F 6C 77 A8 A9 10 :12

P:42 +0 +1 +2 +3 +4 +5 +6 +7 SUM
59A0- 00 00 00 00 00 00 00 E0:E0
59A8- 93 16 56 11 16 16 16 16:68
59B0- 56 11 16 16 E6 FF FF FF:76
59B8- FF FF FF FF FF FF FF FF:F8
59C0- FF FF B3 34 54 35 34 34:D6
59C8- 34 34 54 35 34 34 34 E6:73
59D0- F3 EE FC CF F9 FF FF EE:91
59D8- FC CF F9 FF FF BF B7 BE:F6
59E0- FE AB D5 AA D5 EA 9F BF:45
59E8- B7 FE 9B F3 FF FF FF EE:2E
59F0- FC CF F9 FF FF EE FC E7:93
59F8- 00 00 00 00 00 00 00 00:00
5A00- 83 3C 00 00 1E 00 00 3C:19
5A08- 00 00 00 00 00 00 40 07:47
5A10- 70 01 3C 70 01 00 60 03:81
5A18- 00 3C 1E 00 60 73 01 00:2E
5A20- 60 03 40 3F 1E 00 00 E0:E0
5A28- 93 16 16 16 16 16 16 16:2D
5A30- 16 16 16 16 16 96 B7 CC:87
5A38- 99 B3 E6 C8 99 B3 E6 8C:B8
---------------------------------
SUM  50 E9 7C 9C B0 E4 20 E2 :E7

P:43 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5A40- B7 24 24 34 34 34 34 34:03
5A48- 34 34 34 34 34 34 34 E6:52
5A50- 43 DF 03 55 2A 55 4A DF:22
5A58- 03 15 2A 54 28 65 83 80:26
5A60- 80 80 80 80 80 80 80 80:00
5A68- F0 41 0A 15 2A 54 4A DF:F7
5A70- 03 55 2A 55 4A DF 03 65:68
5A78- 00 00 00 00 00 00 00 00:00
5A80- 83 7C 03 00 7E 7F 7F 7C:FA
5A88- 7F 7F 01 00 00 00 40 3F:7E
5A90- 7E 7F 7F 73 7F 7F 67 1F:73
5A98- 00 00 7E 7F 7F 73 7F 7F:ED
5AA0- 67 03 00 3C 7E 01 00 E0:05
5AA8- 93 16 16 16 16 16 16 16:2D
5AB0- 16 16 16 16 16 96 B7 FE:B9
5AB8- 99 EF 8F D5 F8 FA FF BF:9C
5AC0- B7 24 24 24 34 34 34 34:F3
5AC8- 34 34 34 34 34 34 34 E6:52
5AD0- 03 00 00 00 00 00 00 40:43
5AD8- 20 14 2A 54 28 79 FF AA:FC
---------------------------------
SUM  DB 66 77 D6 5C CE DA 4D :DF

P:44 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5AE0- D5 AA D5 AA D5 AA D5 EA:3C
5AE8- FF 4F 0A 15 2A 14 02 01:AE
```

```
5AF0- 00 00 00 40 00 00 00 60:A0
5AF8- 00 00 00 00 00 00 00 00:00
5B00- 83 7C 03 00 7E 01 00 7C:FD
5B08- 03 00 00 00 00 0F 40 3F:91
5B10- 7E 01 60 73 0F 00 60 1F:E0
5B18- 00 3C 7E 01 60 73 0F 00:9D
5B20- 60 03 00 3C 7E 01 00 E0:FE
5B28- 83 80 00 80 00 80 00 80:83
5B30- 00 80 00 80 00 80 B7 FE:35
5B38- 99 BB A2 D1 A8 EE FF BF:1B
5B40- B7 00 00 00 00 80 80 80:37
5B48- 80 80 80 80 80 80 80 E0:60
5B50- 23 45 0A 15 2A 54 28 51:7E
5B58- 2A 55 42 7F FF AA D5 AA:68
5B60- D5 AA D5 AA D5 AA D5 AA:FC
5B68- D5 EA FF 7F 21 55 2A 15:F2
5B70- 2A 54 28 51 22 45 0A 65:CD
5B78- 00 00 00 00 00 00 00 00:00
---------------------------------
SUM  AC 72 2A 0E D3 72 42 C1 :9E

P:45 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5B80- 83 00 00 00 00 00 00 00:83
5B88- 00 00 00 00 00 00 00 00:00
5B90- 00 00 00 00 00 00 00 00:00
5B98- 00 00 00 00 00 00 00 00:00
5BA0- 00 00 00 00 00 00 00 E0:E0
5BA8- C3 EE E4 CC 88 91 E2 EE:4A
5BB0- E4 CC 88 91 E2 8C B7 FE:EC
5BB8- 99 BB A2 D1 A8 EE FF BF:1B
5BC0- B7 CC 99 91 A2 C4 99 EE:9A
5BC8- E4 CC 88 91 E2 EE 98 E1:12
5BD0- 03 00 00 00 00 00 00 7F:82
5BD8- 7F 7F DF AA D5 AA D5 AA:85
5BE0- D5 AA D5 AA D5 AA D5 AA:FC
5BE8- D5 AA D5 AA FF 7F 7F 7F:7A
5BF0- 00 00 00 00 00 00 00 60:60
5BF8- 00 00 00 00 00 00 00 00:00
5C00- 83 00 00 00 00 0C 78 78:7F
5C08- 3C 40 01 03 03 0F 18 67:11
5C10- 01 00 00 60 40 01 33 70:45
5C18- 40 03 07 1C 38 00 0E 0F:BB
---------------------------------
SUM  8A 23 C0 CD BA AC C3 6A :CD

P:46 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5C20- 18 00 54 00 00 00 00 E0:4C
5C28- D3 AA D5 AA D5 AA D5 AA:FA
5C30- D5 AA D5 AA D5 AA D5 AA:FC
5C38- D5 AA D5 AA D5 AA D5 AA:FC
5C40- D5 AA D5 AA D5 AA D5 AA:FC
5C48- D5 AA D5 AA D5 AA D5 E2:34
5C50- F3 A6 FC BF 82 C0 E2 A6:1E
5C58- FC BF 82 C0 E2 BF 93 FE:2F
5C60- E6 BB B2 D9 AC EE FF BF:84
5C68- 93 FE FF 84 00 C7 FF A6:80
5C70- FC BF 82 C0 E2 A6 FC E7:68
5C78- 00 00 00 00 00 00 00 00:00
5C80- 83 00 00 00 00 0C 18 00:A7
5C88- 0E 1C 38 00 38 0C 00 38:DE
5C90- 00 00 00 00 06 3C 1E 30:90
5C98- 00 03 06 0C 18 00 00 0C:39
5CA0- 00 60 00 00 00 00 00 E0:40
5CA8- D3 AA D5 D5 D2 AA D5 AA:22
5CB0- D5 D5 CA AA D5 AA FF FF:9B
5CB8- FF FF FF FF FF FF FF FF:F8
---------------------------------
SUM  DB 2C 0A 78 17 D3 A1 56 :6A

P:47 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5CC0- FF AA D5 AA D5 D5 D2 AA:4E
5CC8- D5 AA D5 D5 CA AA D5 E2:54
5CD0- F3 A6 FC BF FE FF CF A6:C6
5CD8- FC BF FE FF CF BF 93 FE:D7
5CE0- E6 CB D5 AA D5 E9 FF BF:AC
5CE8- 93 FE EF FF FF 9F FF A6:C2
5CF0- FC BF FE FF CF A6 EC E7:10
5CF8- 00 00 00 00 00 00 00 00:00
5D00- 83 00 00 00 00 0C 18 47:EE
5D08- 7F 47 43 41 41 0F 18 07:B9
5D10- 1C 00 00 00 30 60 70 60:7C
5D18- 40 41 07 3C 30 00 4F 0F:52
5D20- 18 00 6E 00 00 00 00 E0:66
5D28- D3 AA D5 A9 D5 AA D5 AA:F9
5D30- D5 A9 D5 AA E5 FB F7 FF:D3
5D38- F7 FF F7 FF F7 FF F7 FF:D8
5D40- F7 EF D3 AA D5 A9 D5 AA:60
5D48- D5 AA D5 A9 D5 AA D5 E2:33
5D50- F3 A6 FC BF E6 FF FF A6:DE
5D58- FC BF E6 FF FF BF 93 FE:EF
```

```
-----------------------------------
SUM  08 19 49 C5 F0 9B F1 F1 :9C

P:48 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5D60- 86 80 80 80 80 80 F8 BF:BD
5D68- 93 FE EF CC FF FF FF A6:EF
5D70- FC BF E6 FF FF A6 FC E7:28
5D78- 00 00 00 00 00 00 00 00:00
5D80- 83 00 00 00 00 00 00 00:83
5D88- 00 00 00 00 00 00 00 00:00
5D90- 00 00 00 00 00 00 00 00:00
5D98- 00 00 00 00 00 00 00 00:00
5DA0- 00 00 00 00 00 00 00 E0:E0
5DA8- 83 80 60 03 80 80 80 80:66
5DB0- 60 03 80 80 80 80 80 80:63
5DB8- 80 80 80 80 80 80 80 80:00
5DC0- 80 80 80 00 60 03 00 00:E3
5DC8- 00 00 60 03 00 00 00 E0:43
5DD0- F3 A6 FC B3 FE FF FF A6:EA
5DD8- FC B3 FE FF FF BF 93 BE:BB
5DE0- FE AB D5 AA D5 EA 9F BF:45
5DE8- 93 FE E7 FC FF FF FF A6:17
5DF0- FC B3 FE FF FF A6 FC E7:34
5DF8- 00 00 00 00 00 00 00 00:00
-----------------------------------
SUM  F7 75 49 A8 2E F5 9F 3C :5E

P:49 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5E00- 83 3C 00 00 1E 00 00 3C:19
5E08- 00 00 00 00 00 00 40 07:47
5E10- 70 01 3C 70 01 00 60 03:81
5E18- 00 3C 1E 00 60 73 01 00:2E
5E20- 60 03 40 3F 1E 00 00 E0:E0
5E28- 83 80 80 80 80 80 80 80:03
5E30- 80 80 80 80 80 80 93 B3:46
5E38- E6 CC 99 A2 E6 CC 99 B3:EB
5E40- 93 80 80 00 00 00 00 00:93
5E48- 00 00 00 00 00 00 00 E0:E0
5E50- 03 80 20 55 2A 55 0A 80:01
5E58- 20 15 00 00 28 05 F0 FF:51
5E60- D5 AA D5 AA D5 AA D5 FF:51
5E68- 83 50 0A 00 00 54 0A 80:BB
5E70- 20 55 2A 55 0A 80 20 65:03
5E78- 00 00 00 00 00 00 00 00:00
5E80- 83 7C 03 00 7E 7F 3F 7C:BA
5E88- 7F 7F 00 00 00 00 40 3F:7D
5E90- 7E 7F 7F 73 7F 7F 63 1F:6F
5E98- 00 00 7E 7F 7F 73 7F 7F:ED
-----------------------------------
SUM  EA 26 DC 97 30 88 A7 A8 :8A

P:50 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5EA0- 63 03 00 3C 7E 01 00 E0:01
5EA8- 83 80 80 80 80 80 80 80:03
5EB0- 80 80 80 80 80 80 93 FE:91
5EB8- E6 EF A3 D5 E2 FB FF BF:E8
5EC0- 93 80 80 80 00 00 00 00:13
5EC8- 00 00 00 00 00 00 00 E0:E0
5ED0- 23 44 08 11 22 44 08 11:FF
5ED8- 28 14 2A 54 28 79 FF AA:04
5EE0- D5 AA D5 AA D5 AA D5 EA:3C
5EE8- FF 4F 0A 15 2A 14 0A 10:C5
5EF0- 22 44 08 11 22 44 08 61:4E
5EF8- 00 00 00 00 00 00 00 00:00
5F00- 83 7C 7F 1F 7E 7F 3F 7C:55
5F08- 7F 7F 00 00 00 7F 7F 3F:3B
5F10- 7E 01 60 73 7F 7F 63 7F:32
5F18- 7F 3F 7E 01 60 73 7F 7F:0E
5F20- 63 7F 7F 3F 7E 7F 0F E0:8C
5F28- D3 AA D5 AA D5 AA D5 AA:FA
5F30- D5 AA D5 AA D5 AA 93 FE:0E
5F38- E6 BB B2 D9 AC EE FF BF:84
-----------------------------------
SUM  10 D0 74 C5 FC 6C 16 13 :AA

P:51 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5F40- 93 AA D5 AA D5 AA D5 AA:BA
5F48- D5 AA D5 AA D5 AA D5 E2:34
5F50- 23 45 0A 15 2A 54 28 51:7E
5F58- 2A 55 7C 7F DF AA D5 AA:82
5F60- D5 AA D5 AA D5 AA D5 AA:FC
5F68- D5 AA FF 7F 1F 55 2A 15:B0
5F70- 2A 54 28 51 22 45 0A 65:CD
5F78- 00 00 00 00 00 00 00 00:00
5F80- 83 00 00 00 00 00 00 00:83
5F88- 00 00 00 00 00 00 00 00:00
5F90- 00 00 00 00 00 00 00 00:00
5F98- 00 00 00 00 00 00 00 00:00
5FA0- 00 00 00 00 00 00 00 E0:E0
5FA8- B3 A6 98 B3 A2 C4 90 A6:40
```

```
5FB0- 98 B3 A2 C4 90 B3 93 FE:85
5FB8- E6 BB B2 D9 AC EE FF BF:84
5FC0- 93 B2 E6 C4 88 C1 E6 A6:C4
5FC8- 98 B3 A2 C4 90 A6 E4 E6:B1
5FD0- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5FD8- 7F 7F D7 AA D5 AA D5 AA:7D
---------------------------------
SUM  66 0D F6 63 13 8B F0 A3 :FD
P:52 +0 +1 +2 +3 +4 +5 +6 +7 SUM
5FE0- D5 AA D5 AA D5 AA D5 AA:FC
5FE8- D5 AA D5 AA FD 7F 7F 7F:78
5FF0- 7F 7F 7F 7F 7F 7F 7F 7F:F8
5FF8- 03 81 03 8D 02 80 03 80:19
---------------------------------
SUM  2C 54 2C 60 53 28 D6 28 :85
```

리스트 10.9

```
$800,8A5
P:1  +0 +1 +2 +3 +4 +5 +6 +7 SUM
0800- A9 00 8D 00 10 EE 03 08:3F
0808- D0 F8 EE 04 08 AD 04 08:7B
0810- C9 90 D0 EC A9 00 8D 03:4E
0818- 08 A9 10 8D 04 08 A0 0A:04
0820- 88 D0 FD AD 60 C0 29 80:CB
0828- CD A0 09 8D A0 09 D0 05:81
0830- E8 E0 7F D0 E9 BA A2 00:2C
0838- 0D A0 09 8D 00 10 EE 3C:7D
0840- 08 D0 DB EE 3D 08 AD 3D:D0
0848- 08 C9 90 D0 D1 A9 00 8D:38
0850- 3C 08 A9 10 8D 3D 08 60:2F
0858- 00 00 00 00 00 00 00 00:00
0860- AD 01 10 29 7F AA AD 01:BE
0868- 10 29 80 CD A0 09 8D A0:5C
0870- 09 F0 03 8D 30 C0 A0 0F:28
0878- 88 D0 FD CA 30 02 D0 F6:17
0880- EE 61 08 EE 67 08 D0 D8:5C
0888- EE 62 08 EE 68 08 AD 68:CB
0890- 08 C9 90 D0 CB A9 01 8D:33
0898- 61 08 8D 67 08 A9 10 8D:AB
---------------------------------
SUM  73 40 BA 42 6A CB AA 08 :96
P:2  +0 +1 +2 +3 +4 +5 +6 +7 SUM
08A0- 62 08 8D 68 08 60      :C7
---------------------------------
SUM  62 08 8D 68 08 60 00 00 :C7
```

리스트 10.10

```
$1400,1FFF
P:1  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1400- 4C 0D 14 F4 E1 14 00 00:56
1408- 00 00 00 00 00 A9 BE 85:EC
1410- 06 A9 14 85 07 AD 10 C0:CC
1418- AD 00 C0 30 57 AD 61 C0:C2
1420- 4D 62 C0 30 4F A0 00 B1:3F
1428- 06 30 03 4C DD FB C9 FF:25
1430- F0 2A C9 EF B0 42 29 7F:6C
1438- AA CA C8 B1 06 9D FE 1A:A8
1440- C8 CA 10 F7 4C 4A 14 4C:8F
1448- 18 14 18 98 65 06 85 06:D2
1450- A9 00 65 07 85 07 20 0C:CD
1458- 1B 4C 18 14 AD A1 14 F0:E5
1460- 13 85 07 AD A0 14 85 06:8B
1468- A9 00 8D A1 14 AC A2 14:4D
1470- C8 4C 27 14 8D 10 C0 60:0C
1478- 8D 03 14 A6 06 8E A0 14:92
1480- 8E 04 14 A6 07 8E A1 14:96
1488- 8E 05 14 8C A2 14 29 0F:21
1490- AA BD AE 14 85 06 BD AF:20
1498- 14 85 07 A0 01 4C 18 14:B9
---------------------------------
SUM  7B 85 8D 5D 7A DB 12 10 :61
P:2  +0 +1 +2 +3 +4 +5 +6 +7 SUM
14A0- E1 14 00 A9 00 A2 06 9D:E3
14A8- FE 1A CA 10 FA 60 48 15:A9
14B0- 75 15 2A 16 38 17 28 18:59
14B8- 72 18 CD 18 A1 19 8A 00:B3
14C0- 01 01 00 02 03 00 00 00:07
```

```
14C8- 01 F0 8A 00 01 01 00 02:7F
14D0- 06 00 00 00 01 F2 8A 00:83
14D8- 01 01 01 03 02 00 00 00:08
14E0- 01 F4 8A 00 01 01 01 02:84
14E8- 05 00 00 00 01 F6 8A 01:87
14F0- 01 01 04 03 02 00 00 00:0B
14F8- 01 F6 8A 00 01 01 02 03:88
1500- 01 00 00 00 01 F8 8A 01:85
1508- 01 01 02 03 01 00 00 00:08
1510- 01 FA 8A 01 01 01 03 04:8F
1518- 02 00 00 00 01 FC 8A 01:8A
1520- 01 01 03 04 02 00 00 00:0B
1528- 01 F4 8A 01 01 01 05 02:89
1530- 04 00 00 00 01 FE 8A 01:8E
1538- 01 01 05 04 02 00 00 00:0D
---------------------------------
SUM  E3 29 82 FC E9 11 BD DB :1C
P:3  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1540- 01 F4 84 F0 50 1E 50 FF:26
1548- 83 60 3C 14 83 50 36 0A:46
1550- 83 3C 30 32 83 3C 30 0A:1A
1558- 83 40 36 0A 83 48 30 0A:08
1560- 83 40 36 0A 83 3C 30 1E:10
1568- 83 50 30 14 83 59 36 0A:33
1570- 83 60 3C 3C FF 83 78 3C:91
1578- 14 82 28 0A 82 1E 1E 83:09
1580- 50 1E 14 82 00 01 82 1E:A5
1588- 09 82 20 0A 82 24 0A 82:E7
1590- 20 0A 83 78 1E 3C 82 28:29
1598- 1E 83 A0 28 1E 83 78 3C:BE
15A0- 14 82 28 0A 82 1E 1E 83:09
15A8- A0 1E 0A 82 1B 0A 82 1E:0F
15B0- 0A 82 20 0A 82 1B 0A 82:DF
15B8- 1E 0A 83 83 24 1E 83 78:9B
15C0- 24 0A 83 60 24 0A 83 5A:1C
15C8- 24 0A 83 48 2D 1E 83 40:07
15D0- 2D 14 82 36 0A 83 78 3C:3A
15D8- 14 82 28 0A 82 1E 1E 83:09
---------------------------------
SUM  23 45 CC 01 3E 36 31 FC :D6
P:4  +0 +1 +2 +3 +4 +5 +6 +7 SUM
15E0- 50 1E 14 82 00 01 82 1E:A5
15E8- 09 82 20 0A 82 24 0A 82:E7
15F0- 20 0A 83 78 1E 3C 82 28:29
15F8- 1E 83 50 24 14 82 20 0A:D5
```

```
1600- 83 78 1E 14 82 28 0A 83:64
1608- 60 30 1E 83 50 30 14 82:47
1610- 36 0A 83 40 30 14 82 2D:F6
1618- 0A 83 3C 3C 1E 83 50 3C:32
1620- 14 83 60 3C 0A 83 78 3C:74
1628- 3C FF 84 F0 60 3C 14 82:E1
1630- 28 0A 83 5A 1E 14 83 50:14
1638- 1E 0A 84 A0 48 1E 14 82:48
1640- 00 01 82 1E 09 83 40 20:8D
1648- 0A 82 24 0A 82 20 0A 84:EA
1650- F0 50 1E 1E 83 48 1E 14:79
1658- 83 40 1E 0A 83 3C 28 1E:F0
1660- 84 A0 50 28 0A 83 59 28:AA
1668- 0A 83 60 28 0A 84 F0 50:E3
1670- 3C 0A 83 59 3C 0A 83 60:4B
1678- 28 0A 83 6C 1E 14 83 59:2F
---------------------------------
SUM  BF 42 85 C6 A3 0F 20 D7 :F5
P:5  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1680- 1E 0A 84 A0 60 1E 0A 82:56
1688- 1B 0A 82 1E 0A 83 59 20:CB
1690- 0A 82 1B 0A 82 1E 0A 84:DF
1698- B3 48 24 1E 83 60 24 0A:4E
16A0- 83 59 24 0A 83 50 24 0A:0B
16A8- 84 8E 48 2D 1E 84 80 50:F9
16B0- 2D 14 82 36 0A 84 F0 60:D7
16B8- 3C 14 82 28 0A 83 59 1E:FE
16C0- 14 83 50 1E 0A 84 A0 48:7B
16C8- 1E 14 82 00 01 82 1E 09:5E
16D0- 83 40 20 0A 82 24 0A 82:1F
16D8- 20 0A 84 F0 50 1E 1E 83:AD
16E0- 48 1E 14 83 40 1E 0A 83:E8
16E8- 3C 28 1E 84 A0 50 24 0A:24
16F0- 83 59 24 0A 83 60 20 0A:17
16F8- 84 F0 50 1E 0A 83 59 1E:E6
1700- 0A 83 60 28 0A 84 C0 6C:CF
1708- 30 0A 83 60 30 0A 83 59:33
1710- 30 0A 84 A0 50 30 14 82:74
1718- 36 0A 84 80 48 30 14 83:53
---------------------------------
SUM  66 FE BC 6A 40 81 76 DD :9E
P:6  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1720- 40 20 0A 84 F0 3C 3C 1E:81
1728- 84 A0 50 3C 14 84 C0 59:61
1730- 3C 0A 84 F0 60 3C 3C FF:91
```

```
1738- 84 F0 78 1E 14 82 24 0A:CE
1740- 82 28 14 84 B3 B3 28 0A:DA
1748- 84 C0 A0 28 14 82 2D 0A:D9
1750- 82 28 14 82 24 0A 84 A0:92
1758- D8 30 1E 84 D8 80 30 0A:3C
1760- 83 8E 30 0A 83 80 36 0A:8E
1768- 84 F0 78 3C 1E 82 3C 0A:0E
1770- 84 78 C0 3C 0A 83 A0 3C:61
1778- 0A 82 3C 1E 82 30 14 82:2E
1780- 2D 0A 83 B3 28 1E 82 2D:62
1788- 14 82 24 0A 83 C0 28 1E:4D
1790- 84 C0 A0 28 1E 84 A0 D8:26
1798- 28 1E 84 D8 80 28 0A 83:D7
17A0- 8E 28 0A 83 80 28 0A 84:79
17A8- F0 78 1E 14 82 24 0A 82:CC
17B0- 28 14 84 B2 B2 28 0A 84:DA
17B8- C0 A0 28 14 82 2D 0A 82:D7
---------------------------------
SUM  CC 3D 7F 3A E7 1D 07 C2 :8F
P:7  +0 +1 +2 +3 +4 +5 +6 +7 SUM
17C0- 28 14 82 24 0A 84 A0 D8:E8
17C8- 30 1E 84 D8 80 30 0A 83:E7
17D0- 8E 30 0A 83 80 36 0A 84:8F
17D8- F0 78 3C 1E 84 78 F0 3C:EA
17E0- 0A 83 C0 3C 0A 83 A0 3C:F2
17E8- 0A 84 F0 78 3C 1E 84 B2:86
17F0- B2 28 14 82 2D 0A 84 C0:EB
17F8- A0 30 14 82 36 0A 84 D8:02
1800- 80 30 0A 83 8E 30 0A 83:88
1808- 80 2D 0A 84 F0 78 3C 1E:FD
1810- 84 78 A0 3C 0A 83 C0 3C:61
1818- 0A 83 A0 3C 0A 84 F0 78:5F
1820- 3C 1E 84 A0 A0 3C 1E FF:77
1828- 84 00 78 3C 1E 83 60 30:69
1830- 0F 83 59 2D 0F 83 50 28:22
1838- 1E 83 60 30 0F 83 5A 2D:4A
1840- 0F 83 6C 36 1E 83 50 28:4D
1848- 1E 83 78 3C 1E 83 60 30:86
1850- 0F 83 5A 2D 0F 83 48 24:17
1858- 1E 83 60 30 0F 83 5A 2D:4A
---------------------------------
SUM  11 C1 CB DC FF 97 40 23 :72
P:8  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1860- 0F 83 50 28 1E 83 60 30:3B
1868- 0F 83 5A 2D 0F 83 6C 36:4D
1870- 78 FF 84 78 78 3C 1E 84:C9
1878- 60 60 30 0F 84 5A 5A 2D:64
1880- 0F 84 50 50 28 1E 84 60:5D
1888- 60 30 0F 84 5A 5A 2D 0F:13
1890- 84 6C 6C 36 1E 84 50 50:D4
1898- 28 1E 84 78 78 3C 1E 84:98
18A0- 60 60 30 0F 84 5A 5A 2D:64
18A8- 0F 84 48 48 24 1E 84 50:39
18B0- 50 28 0F 84 60 60 30 0F:0A
18B8- 84 5A 5A 2D 1E 84 60 60:C7
18C0- 30 0F 84 6C 6C 36 0F 84:64
18C8- 78 78 3C 78 FF 84 C0 50:37
18D0- 3C 1E 82 30 0F 84 D6 50:C5
18D8- 2D 0F 84 F0 50 28 1E 82:C8
18E0- 30 0F 83 48 2D 0F 84 D6:A0
18E8- 59 36 1E 82 28 0F 83 50:39
18F0- 28 0F 84 F0 60 3C 0F 83:D9
18F8- 59 3C 0F 83 50 30 0F 83:39
---------------------------------
SUM  6F 4D 88 A7 36 20 B9 18 :12
P:9  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1900- 48 2D 0F 84 B3 59 24 1E:56
1908- 82 30 0F 83 48 2D 0F 84:4C
1910- C0 50 28 1E 82 30 0F 83:9A
1918- 48 2D 0F 84 A0 50 36 1E:4C
1920- 83 48 36 0F 83 40 36 0F:18
1928- 84 D6 50 36 1E 83 48 36:FF
1930- 0F 83 40 36 0F 84 C0 3C:97
1938- 3C 1E 83 3C 30 0F 84 D6:B2
1940- 48 2D 0F 84 F0 50 28 1E:8E
1948- 82 30 0F 83 48 2D 0F 84:4C
1950- D6 59 36 1E 82 28 0F 83:BF
1958- 50 28 0F 84 F0 60 3C 0F:A6
1960- 83 59 3C 0F 83 50 30 0F:39
1968- 83 48 2D 0F 84 B3 59 24:BB
1970- 0F 83 50 24 0F 83 48 28:0B
1978- 0F 83 40 30 0F 84 C0 36:8B
1980- 2D 0F 83 50 2D 0F 83 48:16
1988- 30 0F 83 40 36 0F 83 3C:06
1990- 3C 1E 83 50 3C 0F 84 D6:D2
1998- 60 3C 0F 84 F0 78 3C 3C:0F
---------------------------------
SUM  31 96 92 DF 5B 10 13 F5 :AB
P:10 +0 +1 +2 +3 +4 +5 +6 +7 SUM
19A0- FF 84 F0 3C F0 07 82 A0:C8
```

```
19A8- 08 82 78 07 82 60 08 83:76
19B0- 30 50 0F 83 2D 5A 0F 83:2B
19B8- 28 60 07 82 5A 08 82 60:55
19C0- 07 82 6C 08 83 30 78 0F:37
19C8- 83 2D 8E 0F 84 D8 36 A0:7F
19D0- 07 82 80 08 82 6C 07 82:88
19D8- 60 08 83 28 5A 07 82 60:56
19E0- 08 82 6C 07 82 5A 08 84:65
19E8- F0 3C 60 07 82 5A 08 82:F9
19F0- 60 07 82 6C 08 83 30 78:88
19F8- 0F 83 2D 8E 0F 84 B2 24:B6
1A00- 5A 07 82 50 08 82 48 07:0C
1A08- 82 5A 08 83 30 50 0F 83:79
1A10- 2D 48 0F 84 C0 28 60 07:57
1A18- 82 5A 08 82 50 07 82 60:9F
1A20- 08 83 30 78 0F 83 2D 8E:80
1A28- 0F 84 D8 36 D8 07 82 A0:A2
1A30- 08 82 80 07 82 6C 08 82:89
1A38- 50 0F 82 5A 0F 82 50 07:23
---------------------------------
SUM  B1 D2 A1 7F B7 78 84 E1 :37
$1A40,1FFF
P:1  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1A40- 82 5A 08 82 60 07 82 6C:BB
1A48- 08 82 A0 0F 82 80 0F 84:CE
1A50- F0 3C 78 07 82 A0 08 82:57
1A58- 78 07 82 60 08 83 30 50:6C
1A60- 0F 83 2D 5A 0F 83 28 60:33
1A68- 07 82 5A 08 82 60 07 82:56
1A70- 6C 08 83 30 78 0F 83 2D:5E
1A78- 8E 0F 84 D8 36 A0 07 82:58
1A80- 80 08 82 6C 07 82 60 08:67
1A88- 83 28 5A 07 82 60 08 82:78
1A90- 6C 07 82 5A 08 84 F0 3C:07
1A98- 60 07 82 5A 08 82 60 07:34
1AA0- 82 6C 08 83 30 78 0F 83:B3
1AA8- 2D 8E 0F 84 B2 24 5A 07:85
1AB0- 82 50 08 82 48 07 82 5A:87
1AB8- 08 83 28 60 07 82 5A 08:FE
1AC0- 83 30 50 07 82 60 08 84:78
1AC8- C0 2D 6C 07 82 60 08 82:CC
1AD0- 5A 07 82 6C 08 83 30 78:82
1AD8- 07 82 8E 08 83 36 A0 07:7F
---------------------------------
SUM  AE 2C 23 F4 04 C2 5F 91 :A7
P:2  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1AE0- 82 80 08 84 F0 3C 78 0F:41
1AE8- 82 A0 07 82 C0 08 82 F0:E5
1AF0- 0F 82 78 0F 82 F0 1E 82:2A
1AF8- A0 1E FF 4C 0C 1B 00 24:54
1B00- 50 A0 02 03 01 01 01 00:F8
1B08- 00 00 00 00 AD FF 1A 8D:53
1B10- 97 1B 85 08 AD 00 1B 8D:94
1B18- B7 1B 85 09 AD 01 1B 8D:B6
1B20- D7 1B 85 0A AD 02 1B 8D:D8
1B28- 9A 1B 8D A0 1B AD 03 1B:C8
1B30- 8D BA 1B 8D C0 1B AD 04:7B
1B38- 1B 8D DA 1B 8D E0 1B AD:D2
1B40- 05 1B 8D 87 1B AD 06 1B:1D
1B48- 8D A7 1B AD 07 1B 8D C7:72
1B50- 1B A2 00 C6 08 F0 2F C6:70
1B58- 09 F0 4B C6 0A F0 67 CE:39
1B60- 0B 1B F0 0E E0 00 D0 E9:BD
1B68- EA EA A0 03 88 EA 10 FC:F5
1B70- 30 DF E6 09 E6 0A E6 0A:DE
1B78- CE FE 1A F0 08 A9 F0 8D:04
---------------------------------
SUM  13 49 1C 91 E5 3F 2E 97 :F2
P:3  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1B80- 0B 1B 4C 00 1C 60 A9 01:98
1B88- F0 CD AD 30 C0 A9 01 49:4D
1B90- 00 8D 90 1B D0 09 A9 24:DE
1B98- 38 E9 02 85 08 D0 04 A9:2D
1BA0- 02 85 08 E8 D0 B1 A9 01:A2
1BA8- F0 B1 AD 30 C0 A9 01 49:31
1BB0- 00 8D B0 1B D0 09 A9 50:2A
1BB8- 38 E9 03 85 09 D0 04 A9:2F
1BC0- 03 85 09 E8 D0 95 A9 00:87
1BC8- F0 95 AD 30 C0 A9 01 49:15
1BD0- 00 8D D0 1B D0 09 A9 A0:9A
1BD8- 38 E9 01 85 0A D0 04 A9:2E
1BE0- 01 85 0A E8 4C 5F 1B 85:C3
1BE8- 0E 85 0A A9 20 85 0F 85:7F
1BF0- 0B A2 01 86 59 A0 10 B9:F6
1BF8- 51 1B C0 03 90 02 A9 00:6A
1C00- 6C EB 00 A9 0C 85 EB A9:25
1C08- 1C 85 EC 60 A9 D5 8D AC:A4
1C10- 34 A9 C8 8D AC 38 A9 21:E0
1C18- 85 EB A9 1C 85 EC 4C 49:3B
---------------------------------
```

```
SUM  34 65 AC 0C C2 30 55 6E :06
P:4  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1C20- 1B A9 D4 8D 2C 25 A9 7F:9E
1C28- 85 EB A9 1D 85 EC 4C 49:3C
1C30- 1B A9 91 8D AB 34 A9 D4:3E
1C38- 8D AC 38 A9 D2 8D AC 3C:61
1C40- A9 4B 85 EB A9 1C 85 EC:9A
1C48- 4C 49 1B A9 85 8D 2B 2D:C3
1C50- A9 A9 8D 2B 31 A9 CA 8D:3B
1C58- 2C 21 A9 D2 8D 2C 25 A9:4F
1C60- A9 85 EB A9 1D 85 EC 4C:9C
1C68- 49 1B A9 C5 8D AB 34 A9:E7
1C70- D4 8D AC 34 A9 7F 85 EB:D9
1C78- A9 1C 85 EC 4C 49 1B A9:8F
1C80- 95 8D 2B 29 A9 95 8D 2B:6C
1C88- 2D A9 A5 8D 2B 31 A9 F2:FF
1C90- 85 EB A9 1C 85 EC 4C 49:3B
1C98- 1B A9 85 8D AB 34 A9 D2:30
1CA0- 8D AC 38 A9 AE 85 EB A9:E1
1CA8- 1C 85 EC 4C 49 1B A9 D5:BB
1CB0- 8D 2B 29 A9 D0 8D 2C 25:38
1CB8- A9 17 85 EB A9 1D 85 EC:67
---------------------------------
SUM  C2 CD B1 E7 2D 78 19 77 :5C
P:5  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1CC0- 4C 49 1B A9 C5 8D AC 34:8B
1CC8- A9 A8 8D AC 38 A9 CA 8D:C2
1CD0- AC 3C A9 DD 85 EB A9 1C:A3
1CD8- 85 EC 4C 49 1B A9 D2 8D:29
1CE0- 2C 21 A9 D2 8D 2C 25 A9:4F
1CE8- 50 85 EB A9 1D 85 EC 4C:43
1CF0- 49 1B A9 D5 8D B2 34 A9:FE
1CF8- C8 8D B2 38 A9 07 85 EB:5F
1D00- A9 1D 85 EC 4C 49 1B A9:90
1D08- D4 8D 32 25 A9 65 85 EB:36
1D10- A9 1E 85 EC 4C 49 1B A9:91
1D18- 91 8D B1 34 A9 D4 8D B2:BF
1D20- 38 A9 D2 8D B2 3C A9 31:08
1D28- 85 EB A9 1D 85 EC 4C 49:3C
1D30- 1B A9 85 8D 31 2D A9 A9:86
1D38- 8D 31 31 A9 CA 8D 32 21:42
1D40- A9 D2 8D 32 25 A9 8F 85:1C
1D48- EB A9 1E 85 EC 4C 49 1B:D3
1D50- A9 C5 8D B1 34 A9 D4 8D:EA
1D58- B2 34 A9 65 85 EB A9 1D:2A
---------------------------------
SUM  BE 9E 8B E1 63 6A 28 70 :2D
P:6  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1D60- 85 EC 4C 49 1B A9 95 8D:EC
1D68- 31 29 A9 95 8D 31 2D A9:2C
1D70- A5 8D 31 31 A9 D8 85 EB:85
1D78- A9 1D 85 EC 4C 49 1B A9:90
1D80- 85 8D B1 34 A9 D2 8D B2:B1
1D88- 38 A9 94 85 EB A9 1D 85:30
1D90- EC 4C 49 1B A9 D5 8D 31:D8
1D98- 29 A9 D0 8D 32 25 A9 FD:2C
1DA0- 85 EB A9 1D 85 EC 4C 49:3C
1DA8- 1B A9 C5 8D B2 34 A9 A8:4D
1DB0- 8D B2 38 A9 CA 8D B2 3C:65
1DB8- A9 C3 85 EB A9 1D 85 EC:13
1DC0- 4C 49 1B A9 D2 8D 32 21:0B
1DC8- A9 D2 8D 32 25 A9 36 85:C3
1DD0- EB A9 1E 85 EC 4C 49 1B:D3
1DD8- A9 D5 8D C6 34 A9 C8 8D:03
1DE0- C6 38 A9 ED 85 EB A9 1D:CA
1DE8- 85 EC 4C 49 1B A9 D4 8D:2B
1DF0- 46 25 A9 4B 85 EB A9 1F:97
1DF8- 85 EC 4C 49 1B A9 91 8D:E8
---------------------------------
SUM  4B C1 71 8A 0D 8D 9E EC :2B
P:7  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1E00- C5 34 A9 D4 8D C6 38 A9:AA
1E08- D2 8D C6 3C A9 17 85 EB:91
1E10- A9 1E 85 EC 4C 49 1B A9:91
1E18- 85 8D 45 2D A9 A9 8D 45:A8
1E20- 31 A9 CA 8D 46 21 A9 D2:13
1E28- 8D 46 25 A9 75 85 EB A9:2F
1E30- 1F 85 EC 4C 49 1B A9 C5:AE
1E38- 8D C5 34 A9 D4 8D C6 34:8A
1E40- A9 4B 85 EB A9 1E 85 EC:9C
1E48- 4C 49 1B A9 95 8D 45 29:E9
1E50- A9 95 8D 45 2D A9 A5 8D:18
1E58- 45 31 A9 BE 85 EB A9 1E:14
1E60- 85 EC 4C 49 1B A9 85 8D:DC
1E68- C5 34 A9 D2 8D C6 38 A9:A8
1E70- 7A 85 EB A9 1E 85 EC 4C:6E
1E78- 49 1B A9 D5 8D 45 29 A9:86
1E80- D0 8D 46 25 A9 E3 85 EB:C4
1E88- A9 1E 85 EC 4C 49 1B A9:91
1E90- C5 8D C6 34 A9 A8 8D C6:F0
```

```
1E98- 38 A9 CA 8D C6 3C A9 A9:8C
---------------------------------
SUM  95 A0 02 56 AA A5 28 E4 :E8
P:8  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1EA0- 85 EB A9 1E 85 EC 4C 49:3D
1EA8- 1B A9 D2 8D 46 21 A9 D2:05
1EB0- 8D 46 25 A9 1C 85 EB A9:D6
1EB8- 1F 85 EC 4C 49 1B A9 D5:BE
1EC0- 8D CC 34 A9 C8 8D CC 38:8F
1EC8- A9 D3 85 EB A9 1E 85 EC:24
1ED0- 4C 49 1B A9 D4 8D 4C 25:2B
1ED8- A9 6A 85 EB A9 1C 85 EC:B9
1EE0- 4C 49 1B A9 91 8D CB 34:76
1EE8- A9 D4 8D CC 38 A9 D2 8D:16
1EF0- CC 3C A9 FD 85 EB A9 1E:E5
1EF8- 85 EC 4C 49 18 A9 85 8D:DC
1F00- 4B 2D A9 A9 8D 4B 31 A9:7C
1F08- CA 8D 4C 21 A9 D2 8D 4C:18
1F10- 25 A9 99 85 EB A9 1C 85:21
1F18- EC 4C 49 1B A9 C5 8D CB:62
1F20- 34 A9 D4 8D CC 34 A9 31:18
1F28- 85 EB A9 1F 85 EC 4C 49:3E
1F30- 1B A9 95 8D 4B 29 A9 95:98
1F38- 8D 4B 2D A9 A5 8D 4B 31:5C
---------------------------------
SUM  44 2D 98 CF 92 2C C6 BF :1B
P:9  +0 +1 +2 +3 +4 +5 +6 +7 SUM
1F40- A9 C3 85 EB A9 1C 85 EC:12
1F48- 4C 49 1B A9 85 8D CB 34:6A
1F50- A9 D2 8D CC 38 A9 60 85:9A
1F58- EB A9 1F 95 EC 4C 49 1B:D4
1F60- A9 D5 8D 4B 29 A9 D0 8D:85
1F68- 4C 25 A9 0C 85 EB A9 1C:5B
1F70- 85 EC 4C 49 1B A9 C5 8D:1C
1F78- CC 34 A9 A8 8D CC 38 A9:8B
1F80- CA 8D CC 3C A9 8F 85 EB:07
1F88- A9 1F 85 EC 4C 49 1B A9:92
1F90- D2 8D 4C 21 A9 D2 8D 4C:20
1F98- 25 A9 31 85 EB A9 1C 85:B9
1FA0- EC 4C 49 1B 00 F1 1E 95:40
1FA8- 00 E6 1E D0 02 E6 1F A5:80
1FB0- 01 F1 1E 95 01 8A 69 00:99
1FB8- 85 1D 4C 91 1D A2 18 A5:FB
1FC0- 1E 81 00 E6 18 D0 02 E6:55
1FC8- 19 A5 1F 81 00 E6 18 D0:2C
1FD0- 02 E6 19 18 B1 1E E6 1E:EC
1FD8- D0 02 E6 1F 65 1E AA B1:B5
---------------------------------
SUM  B4 D1 34 AA 7F EF 20 68 :59
P:10 +0 +1 +2 +3 +4 +5 +6 +7 SUM
1FE0- 1E 65 1F 85 1F 86 1E 4C:36
1FE8- 91 1D 00 00 4F 4F 00 00:4C
1FF0- 4F 4F 00 00 4F 4F 00 00:3C
1FF8- 4F 4F 00 00 4F 4F 00 00:3C
---------------------------------
SUM  4D 20 1F 85 0C 73 1E 4C :FA
```

§3. 스크린 스위치

1. 개요

애플 Ⅱe의 특수한 기능으로서는 소프트 스위치와 더블 고해상도 그래픽 등을 들 수 있다. 이 중 소프트 스위치는 다른 기종에서는 전혀 찾아볼 수 없는 특수한 기능이다. 그리고 더블 고해상도 그래픽도 특수한 기능이지만 사용법이 까다롭고 베이직에서는 직접 사용할 수 없다는 단점이 있다.

이 프로그램은 애플 Ⅱe의 장점인 소프트 스위치(soft swith)를 혼합하여 다섯 가지 모드를 손쉽게 조절하며 Full 모드와 Mix 모드(그래픽과 아래 네 줄의 텍스트)를 자유롭게 쓸 수 있게 하였다. 또한 확장성을 고려하여 다른 앰퍼샌드 명령어(& 명령어)도 쓸 수 있게 하였다(즉 다른 &문 프로그램을

실행시킨 후 이 프로그램을 실행시켜도 전에 실행시켰던 프로그램을 같이 사용할 수가 있다).

2. 프로그램 입력

이 프로그램은 DOS Tool Kit 내의 어셈블러로 짜여져 있다. 다른 어셈블러 사용자는 자신의 어셈블러에 맞게 개조하여야 한다. 어셈블러가 없으면 리스트2의 덤프리스트를 입력한 후 다음과 같이 세이브시킨다.

BSAVE SS][, A$ 6000, L$CO

3. 명령어

& MODE 1

베이직의 TEXT와 같다.

& MODE 2

베이직의 GR명령과 비슷하지만 이 명령은 GR 화면을 지우지 않고 스크린 전환만 한다.

& MODE 3

베이직으로는 복잡한 명령들을 조합하여 Double GR 화면을 만든다. 이때도 화면은 지우지 않는다.

& MODE 4

베이직의 HGR명령과 비슷하지만 HGR 화면은 지우지 않는다.

MODE 5

더블 고해상도 그래픽을 세트한다.

& MODE 6

각각의 화면 모드를 세트한 후에 이 명령을 사용하면 전체가 같은 모드로 된다. 즉 & MODE 2를 한 후 이 명령을 쓰면 전체가 GR화면이 된다.

& MODE 7

각각의 화면 모드를 세트한 후에 이 명령을 사용하면 네 줄의 텍스트가 혼합된 화면이 나온다.

4. 주의 사항

이 프로그램의 명령어 중 &MODE 3.5는 PR#3을 한 후에 사용한다. 만약 40컬럼 모드에서 사용하

면 명령 실행 후 이상하게 되어버린다.

5. 프로그램 설명

22~32행 : & 명령어 세트.
40~51행 : 명령어가 맞는지 검사.
57행 : 만약 틀린 명령어일 때 이전에 세트한 & 명령 실행 번지로 간다.
63~78행 : 숫자에 따른 분기.
84~끝행 : 각 모드의 실행.

리스트 10.11

```
SOURCE FILE: PGM
0000:                1 ;-------------------------
0000:                2 ;[                        ]
0000:                3 ;[   & SCREEN SWITCH      ]
0000:                4 ;[                        ]
0000:                5 ;[    BY AHN DONG HOON    ]
0000:                6 ;[                        ]
0000:                7 ;=========================
0000:                8 ;
0000:                9 ;THIS PROGRAM USE &VEC-
0000:               10 ; TOR ONLY...
0000:               11 ;
0000:               12 ;APPLE'S SUBROUTINES...
0000:               13 ;
FB2F:               14 TEXT      EQU   $FB2F
0000:               15 ;
03F5:               16 AMPER     EQU   $3F5
00B1:               17 CHRGET    EQU   $00B1
5FF0:               18 OLDAMP    EQU   $5FF0
0000:               19 ;
0000:               20 ;
------ NEXT OBJECT FILE NAME IS PGM.OBJ0
6000:               21           ORG   $6000
6000:AD F6 03       22 AMPERSET LDA   AMPER+1
6003:8D F0 5F       23           STA   OLDAMP
6006:AD F7 03       24           LDA   AMPER+2
6009:8D F1 5F       25           STA   OLDAMP+1
600C:A9 4C          26           LDA   #$4C
600E:8D F5 03       27           STA   AMPER
```

```
6011:A9 1C                    LDA   #>CHK
6013:8D F6 03                 STA   AMPER+1
6016:A9 60                    LDA   #<CHK
6018:8D F7 03                 STA   AMPER+2
601B:60                       RTS
601C:               ;
601C:               ;START MAIN...
601C:               ;
601C:               ;------------------
601C:               ; CHECK COMMAND  ]
601C:               ;------------------
601C:               ;
601C:C9 4D          CHK       CMP   #$4D
601E:D0 17                    BNE   ERROR
6020:20 B1 00                 JSR   CHRGET
6023:C9 4F                    CMP   #$4F
6025:D0 10                    BNE   ERROR
6027:20 B1 00                 JSR   CHRGET
602A:C9 44                    CMP   #$44
602C:D0 09                    BNE   ERROR
602E:20 B1 00                 JSR   CHRGET
6031:C9 45                    CMP   #$45
6033:F0 05                    BEQ   L1
6035:D0 00                    BNE   ERROR
6037:               ;
6037:               ;------------------
6037:               ; ERROR OPERATING]
6037:               ;------------------
6037:               ;
6037:6C F0 5F       ERROR     JMP   (OLDAMP)
603A:               ;
603A:               ;------------------
603A:               ; BRANCH EACH...  ]
603A:               ;------------------
603A:               ;
603A:20 B1 00       L1        JSR   CHRGET
603D:C9 31                    CMP   #$31
603F:F0 1B                    BEQ   M1
6041:C9 32                    CMP   #$32
6043:F0 1D                    BEQ   M2
6045:C9 33                    CMP   #$33
6047:F0 28                    BEQ   M3
6049:C9 34                    CMP   #$34
604B:F0 36                    BEQ   M4
604D:C9 35                    CMP   #$35
```

```
604F:F0 41              BEQ   M5
6051:C9 36              CMP   #$36
6053:F0 4F              BEQ   M6
6055:C9 37              CMP   #$37
6057:F0 51              BEQ   M7
6059:4C 37 60           JMP   ERROR
605C:               ;
605C:               ;-----------------
605C:               ;  MODE 1:TEXT  ]
605C:               ;-----------------
605C:               ;
605C:20 2F FB       M1          JSR   TEXT
605F:4C B1 00           JMP   CHRGET
6062:               ;
6062:               ;-----------------
6062:               ;  MODE 2: G R  ]
6062:               ;-----------------
6062:               ;
6062:8D 50 C0       M2          STA   $C050
6065:8D 53 C0           STA   $C053
6068:8D 54 C0           STA   $C054
606B:8D 56 C0           STA   $C056
606E:4C B1 00           JMP   CHRGET
6071:               ;
6071:               ;-----------------
6071:               ;  MODE 3: DGR  ]
6071:               ;-----------------
6071:               ;
6071:8D 50 C0       M3          STA   $C050
6074:8D 56 C0           STA   $C056
6077:8D 53 C0           STA   $C053
607A:8D 0D C0           STA   $C00D
607D:8D 5E C0           STA   $C05E
6080:4C B1 00           JMP   CHRGET
6083:               ;
6083:               ;-----------------
6083:               ;  MODE 4: HGR  ]
6083:               ;-----------------
6083:               ;
6083:8D 50 C0       M4          STA   $C050
6086:8D 53 C0           STA   $C053
6089:8D 54 C0           STA   $C054
608C:8D 57 C0           STA   $C057
608F:4C B1 00           JMP   CHRGET
6092:               ;
```

```
6092:                 ;------------------
6092:                 ;   MODE 5: DHGR ]
6092:                 ;------------------
6092:                 ;
6092:8D 50 C0         M5        STA   $C050
6095:8D 57 C0                   STA   $C057
6098:8D 53 C0                   STA   $C053
609B:8D 0D C0                   STA   $C00D
609E:8D 5E C0                   STA   $C05E
60A1:4C B1 00                   JMP   CHRGET
60A4:                 ;
60A4:                 ;------------------
60A4:                 ;   MODE 6: FULL ]
60A4:                 ;------------------
60A4:                 ;
60A4:8D 52 C0         M6        STA   $C052
60A7:4C B1 00                   JMP   CHRGET
60AA:                 ;
60AA:                 ;------------------
60AA:                 ;   MODE 7: MIX  ]
60AA:                 ;------------------
60AA:                 ;
60AA:8D 53 C0         M7        STA   $C053
60AD:4C B1 00                   JMP   CHRGET
60B0:                 ;
60B0:                 ;------------------
60B0:                 ;                 ]
60B0:                 ;  GOOD - BYE !   ]
60B0:                 ;                 ]
60B0:                 ;------------------
60B0:                 ;
```

리스트 10.12

```
6000- AD F6 03 8D F0 5F AD F7
6008- 03 8D F1 5F A9 4C 8D F5
6010- 03 A9 1C 8D F6 03 A9 60
6018- 8D F7 03 60 C9 4D D0 17
6020- 20 B1 00 C9 4F D0 10 20
6028- B1 00 C9 44 D0 09 20 B1
6030- 00 C9 45 F0 05 D0 00 6C
6038- F0 5F 20 B1 00 C9 31 F0
6040- 1B C9 32 F0 1D C9 33 F0
6048- 28 C9 34 F0 36 C9 35 F0
```

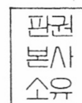
```
6050- 41 C9 36 F0 4F C9 37 F0
6058- 51 4C 37 60 20 2F FB 4C
6060- B1 00 8D 50 C0 8D 53 C0
6068- 8D 54 C0 8D 56 C0 4C B1
6070- 00 8D 50 C0 8D 56 C0 8D
6078- 53 C0 8D 0D C0 8D 5E C0
6080- 4C B1 00 8D 50 C0 8D 53
6088- C0 8D 54 C0 8D 57 C0 4C
6090- B1 00 8D 50 C0 8D 57 C0
6098- 8D 53 C0 8D 0D C0 8D 5E
60A0- C0 4C B1 00 8D 52 C0 4C
60A8- B1 00 8D 53 C0 4C B1 00
60B0- 0D 45 53 43 0D 0D 00 C9
60B8- 60 8F 02 B9 30 2C 30 3A
60C0- B9
```

글쓴이들
(컴퓨터학습 PC클럽 애플팀)
윤 은식(한양대 전기 1)
유 한석(한양대 경제 1)
이 재철(인천고 3)
손 재철(숭실고 3)
최 영익(성현고 3)
권 구철(성보고 2)
안 동훈(용현중 3)
이 삼구(용현중 3)

애플 IIe 테크노트 정가 5,800 원

1989년 7월 25일 인쇄
1989년 8월 1일 발행

발행인 민 경 현
인쇄처 영신인쇄공사
발행처 민 컴



서울시 마포구 서교동 393-4
121-210
☎ 333-4101~9
(등록년월일 1964. 10. 9 제 2-1115호)